VŠB - Technická univerzita Ostrava

Fakulta strojní

Katedra automatizační techniky a řízení

Monitorování reálného objektu nástroji SPC v prostředí InTouch

Monitoring a Real Time Object with SPC Tools in InTouch

Student: Bc. Petr Kopečný

Vedoucí diplomové práce: doc. Ing. Lenka Landryová, CSc.

Ostrava: 2011

VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Zadání diplomové práce

Student: **Bc. Petr Kopečný**

Studijní program: N2301 Strojní inženýrství

Studijní obor: 3902T004 Automatické řízení a inženýrská informatika

Téma: Monitorování reálného objektu nástroji SPC v prostředí InTouch
Monitoring a Real Time Object with SPC Tools in InTouch

Zásady pro vypracování:

1. Seznamte se s koncepcí vývoje aplikací určených pro monitorování a řízení technických prostředků, provozovaných zařízení a oblastí průmyslového provozu na úrovni MES systémů pro malé a střední podniky.
2. Popište vybrané vývojové a runtime softwarové prostředí, uveďte přehled vhodných nástrojů pro vývoj a konfigurace objektů aplikace monitorování a supervizního řízení.
3. Popište model vybraného dílčího zařízení včetně typů, alarmních limitů a způsobu ukládání historie vámi navrhovaných proměnných pro monitorované veličiny tohoto modelu.
4. Vytvořte demonstrační aplikaci se simulovanými veličinami a seznamte se s vybranými SPC nástroji, s jejich funkcemi při vývoji a konfiguraci objektů aplikace monitorování a supervizního řízení.
5. Analyzujte vybraný model v laboratoři z hlediska posuzování jeho funkčnosti a monitorovaných veličin.
6. Testujte aplikaci s ohledem použití nástrojů SPC v prostředí InTouch a zhodnoťte dosažené výsledky.

Seznam doporučené odborné literatury:

[1] AVERY, J. *ASP.NET Konfigurace a nastavení*, 1. vydání. Praha: BEN, 2004, 194 s., ISBN 80-251-0121-5
[2] LANDRYOVÁ, L. Knowledge Management and Life Cycle Support through Open System Architecture. In *Proceedings of the IFIP WG 5.7 Working Conference on Human Aspects In Production Management. European Series in Industrial Management*, edited by prof. Gert Zulch, Volume 2–2003, Current Trends In Production Management, Shaker Verlag, pg 260–266, ISBN 3–8322-1935–8, ISSN 1437–7675
[3] VLACH, J. *Řízení a vizualizace technologických procesů*, 1. vydání. Praha: BEN, 2002, 160 s., ISBN 80-86056-66-X
[4] WONDERWARE. Help k programu Factory Suite 2000/InTouch 9.5/IAS [CD ROM]

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Lenka Landryová, CSc.**

Datum zadání: 17.12.2010
Datum odevzdání: 23.05.2011

prof. Ing. Jiří Tůma, CSc.
vedoucí katedry

prof. Ing. Radim Farana, CSc.
děkan fakulty

Prohlášení diplomanta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě 22.5.2011 .....

.....
Podpis studenta

Prohlašuji, že

- jsem byl seznámen s tím, že na moji diplomovou (bakalářskou) práci se plně vztahuje zákon č. 121/2000 Sb., autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB-TUO“) má právo nevýdělečně ke své vnitřní potřebě diplomovou (bakalářskou) práci užít (§ 35 odst. 3).
- souhlasím s tím, že diplomová (bakalářská) práce bude v elektronické podobě uložena v Ústřední knihovně VŠB-TUO k nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové (bakalářské) práce. Souhlasím s tím, že údaje o kvalifikační práci budou zveřejněny v informačním systému VŠB-TUO.
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – diplomovou (bakalářskou) práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě : 22. 5. 2011 ..... Kopečný

podpis

Jméno a příjmení autora práce: Bc. Petr Kopečný

Adresa trvalého pobytu autora práce: Antonína Poledníka 4/8
Ostrava - Dubina
700 30

Anotace

KOPEČNÝ, P. *Monitorování reálného objektu nástroji SPC v prostředí InTouch.* Ostrava: Katedra ATŘ-352 VŠB-TUO, 2011. 58 s. Diplomová práce, vedoucí: doc. Ing. Lenka Landryová, CSc.

Po úvodní kapitole, se druhá část zabývá teorií rozdílu mezi dvěma přístupy k vývoji aplikací určených pro monitorování a řízení technických prostředků. Ukazuje jejich rozdíly a použitelnost. Diplomová práce pokračuje popisem vybraného softwarového prostředí, tedy InTouch, a jeho možnosti a nástroje pro vývoj a konfiguraci. Další část uvádí laboratorní model v laboratoři. Jeho schéma a hardwarové a softwarové propojení. Následující kapitola pojednává o SPC nástrojích a jejich vlastnostech a možnostech. Nakonec jsou uvedeny dvě aplikace v prostředí InTouch. Aplikaci se simulovanými veličinami a aplikaci napojenou na reálný model s funkčními SPC nástroji.

Annotation

KOPEČNÝ, P. *Monitoring of real object by SPC tools in InTouch environment.* Ostrava: Department of Control System and Instrumentation, VŠB – Technical University of Ostrava, 2011. 58 p. Thesis, head: doc. Ing. Lenka Landryová, CSc.

After the introductory chapter, the second part deals with the theory of the difference between the two approaches to developing applications for monitoring and managing technical resources. It shows their differences and usability. The thesis goes on to describe the selected software environment, so InTouch and its capabilities and tools for development and configuration. Furthermore another section shows a laboratory model in the laboratory. His scheme, and hardware and software links. The next chapter deals with the SPC tools and their properties and possibilities. Finally there are, two applications in InTouch. Application of the simulated quantities and applications connected to the real model with functional SPC tools.

Obsah

Seznam použitých symbolů

A_2	Statistické konstanty UCL (LCL) diagramu $(\bar{x}, R)$ pro x
A_3	Statistické konstanty UCL (LCL) diagramu $(\bar{x}, s)$ pro x
B_3	Statistické konstanty UCL diagramu $(\bar{x}, s)$ pro s
B_4	Statistické konstanty LCL diagramu $(\bar{x}, s)$ pro s
Cp	Index způsobilosti centrování mezi tolerančním polem
Cpk	Index způsobilosti seřízení na střed tolerančního pole
D_3	Statistické konstanty UCL diagramu $(\bar{x}, R)$ pro R
D_4	Statistické konstanty LCL diagramu $(\bar{x}, R)$ pro R
h	Šířka intervalu
m	Určitý moment přibližného průměru vzorku.
n	Počet použitých vzorků
N	Počet zobrazených vzorků
R	Rozptyl
$\bar{R}$	Průměr rozptylu podskupiny
s	Směrodatná odchylka
$\bar{s}$	Průměr směrodatné odchylky podskupiny
x	Jednotlivá hodnota měření
$\bar{x}$	Průměr hodnot
$\bar{\bar{x}}$	Výběrový průměr
x_{max}	Maximální x
x_{min}	Minimální x
α	Riziko zbytečného signálu
β	Riziko chybějícího signálu

Seznam použitého značení

ADO	ActiveX Data Objects
CL	Control Line
ČSN	České technické Normy
DCS	Distributed Control System
DDE	Dynamic Data Exchange
DTC	Direct Torque Control
ERP	Enterprise Resource Planning
HMI	Human Machine Interface
HR	Human Resource
HW	HardWare
IDE	Integrated Development Environment
ISO	International Organization for Standardization
I/O	Input / Output
IP	Internet Protocol
IT	Informační Technologie
KPI	Key Performance Indicators
LCL	Lower Control Line
LSL	Lower Specification Limit
MCS	Manufacturing Control Systems
MES	Manufacturing Execution Systems
MMI	Man-Machine Interface
MS	MicroSoft
ODBC	Open DataBase Connectivity
OEMs	Original Equipment Manufacturers
OLE	Object Linking and Embedding

OOP	Object-Oriented Programming
OPC	OLE for Process Control
PC	Personal Computer
PCMCIA	Peripheral Component MicroChannel Interconnect Architecture
PLC	Programmable Logic Controller
SCADA	Supervisory Control And Data Acquisition
SPC	Statistical Process Control
SQL	Structured Query Language
SW	SoftWare
UCL	Upper Control Line
USB	Universal Serial Bus
USL	Upper Specification Limit
VARs	Value-Added Resellers

1 Úvod

Nyní, na začátku dvacátého-prvního století, je efektivita, tedy co nejvyšší zisk za použití co nejmenších nákladů, velice důležitá. Ať už je to kvůli finanční krizi nebo pouhé optimalizaci, SPC nástroje (metody řízení kvality) jsou velice důležité. Dovolují nám najít příčiny problémů nebo samotné vady. Dalo by se říct, že se v dnešní době žádná větší společnost bez SPC nástrojů neobejde.

Ale co jsou to vlastně tyto mocné SPC nástroje? Nástroje SPC představují sadu nástrojů pro analýzy průběhu veličin, sledovaných v reálném čase, zejména pak pro jejich kvalitu, odchylky od limitních hodnot, statistické výpočty pro průměr, kvadratické odchylky a jejich zpracování do regulačních diagramů, histogramů, Pareto grafů a dalších.

Pro použití SPC nástrojů se ovšem neobejdeme bez jejich analýzy. A k analýze musíme použít samotnou vizualizaci, která zpracuje příchozí data, uloží je do databáze a poskytne operátorovi pohled ať už na graf nebo jen prosté čísla.

K tomuto účelu se hodí i velice dobré vizualizační protřídí InTouch společnosti Wonderware na kterém jsem vytvořil také svou vlastní vizualizační aplikaci. Spolu s nástroji SPC budu testovat jejich funkčnost a možné použití, které by mohl využít někdo další spolu s nějakým druhem poruchy, například brzdy.

2 Koncepce vývoje aplikací

V dnešní době existují dvě koncepce vývoje aplikací pro monitorování a řízení technických prostředků v oblasti průmyslového provozu na úrovni MES systémů. Jsou to tradiční koncepce vývoje vizualizovaných proměnných, které mohou být definovány pro tyto vývojová prostředí svým názvem, datovým typem, rozsahem, apod. - Tag-based - a novější koncepce Component object-based. Architektura založená na komponentách (a koncepci vývoje Component-based) se už nějakou dobu používá. Také objektově orientovaný (object-oriented) vývoj se využívá. Nicméně pouze nedávno se tyto metody spojily a obohatily svět okolo SCADA a procesního řízení. Jejich hlavní výhodou je především až 80% úspora nákladů ve vývoji a inženýrské práci, takže není divu, že se staly nástupcem předchozích metod. [GARBRECHT, 2006]

2.1 Definice

Následující řádky jsou věnovány definicím, které se často v této problematice vyskytují. Proto následuje jejich krátký popis.

Objektově orientované programování (Object oriented programing)

Objektově orientované programování neboli OOP je model programovacího jazyka orientovaný okolo takzvaných objektů více než okolo akcí, a dat více než logiky. V minulosti byly programy zobrazeny jako logické procedury, které si vzaly vstupní data, zpracovaly je a vytvořily výstupní data. Programování bylo o tom jak psát logiku, a ne jak definovat data. Oproti tomu, objektově orientované programování bere v potaz to, co je opravdu důležité, a to objekty s kterými chceme manipulovat. OOP je založené na těchto koncepcích:

- Objekty - jednotlivé prvky jsou v programu seskupeny do entit, nazývaných objekty.
- Abstrakce - každý objekt pracuje jako černá skříňka, která dokáže provádět určité činnosti a komunikovat s okolím, bez toho aby vyžadovala znalost způsobu, kterým vnitřně pracuje.
- Zapouzdření - zajišťuje vnitřní bezpečnost tak, aby jiné objekty nemanipulovaly s vnitřnostmi dalších.
- Skládání – objekt může obsahovat jiné objekty.
- Delegování - objekt může využívat služeb jiných objektů.

- Dědičnost – objekty mohou přejímat některé vlastnosti jiných objektů orientovaných do stromové struktury.

- Polymorfizmus - objekt se chová podle toho, jaké třídy je instancí.

Komponenta (Component)

V OOP je komponenta znovupoužitelný programový blok, který je možné kombinovat s dalšími bloky tvořícími aplikaci. Komponenty mohou být umístěny třeba i na odlišných serverech a komunikují každý s každým za účelem provádění nějaké služby. Například tlačítko v grafickém uživatelském interface, malý kalkulátor nebo spojení s databází.

Kontejner (Container)

Komponenty běží mimo kontext kontejneru. Kontejner obsahuje například webové servery, aplikační servery nebo databázové servery.

Komponenta založená na objektech (Component object-based)

Tento termín odkazuje na systém, který používá objektově orientovaný vývoj k vytvoření aplikací založených na komponentách. Protože spojení mezi komponentami a vývojovým prostředím (development) s běžícím prostředím (run-time) je udržován, inkrementální změny mohou být provedeny bez vypnutí celého systému.

Replikace (Replication)

Tento termín odkazuje na vytvoření komponent z šablon objektů.

Změna propagace (Change of propagation)

V objektově orientovaném programování tento termín odkazuje na proces změny šablony objektu a pozdější selektivní distribuci, která změní aplikaci, několik aplikací nebo i celý systém. [GARBRECHT D. STEVEN, 2006]

Plant Intelligence

Každý výrobní podnik má bez rozdílů na druhu výroby dva světy. Výrobní a obchodní. Výrobní svět se stará o hladký průběh výrobního procesu a obchodní se stará o otázky finančního rázu. Mezi těmito dvěma světy existuje velmi velká informační bariéra. Řídící pracovníci potom dělají rozhodnutí na základě zastaralých nebo neúplných informací,

případně nemají potřebné informace vůbec k dispozici. Základní úkol je tedy zrušit tuto bariéru, protože pouze díky přesným informacím mohou řídící pracovníci optimalizovat výrobu. A vzhledem k velké konkurenci a často měnícím se požadavkům vzniká potřeba přístupu ke klíčovým informacím v reálném čase. Plant intelligence je název pro koncepci integrace obou světů ve výrobních podnicích do jednoho celku. Cílem je zvýšit efektivitu a kvalitu výroby za účelem zvýšení zisku. V praxi je úkolem Plant Intelligence shromažďovat všechna výrobní i nevýrobní data z nejrůznějších zdrojů (měřících zařízení, relačních databází, různých dokumentů, SCADA, HMI, PLC a DCS) na jedno místo, analyzovat a převádět je na potřebné manažerské informace a poskytovat je tak, aby bylo možné vytvářet správná rozhodnutí. Důležitá je také zajištění zpětné vazby. [ŘÍHA, 2007]

2.2 Tag-based systems

Z vývoje přístupu k datům, skriptů, alarmů, datové analýzy víme, že byla založena na konceptu tagů (tags). Ačkoliv se může zdát tento přístup velmi jednoduchý a přenositelný z jednoho projektu do druhého. Tag-based prostředí ale používá ploché názvosloví, což je velmi krátkozraké, protože individuální elementy nemohou být spojeny nebo organizovány do více inteligentních struktur s vestavěnými vztahy nebo nezávislostmi.

Globální změny do Tag-based databáze jsou vytvářeny externě do vývojového prostředí jako textový nebo tabulkový soubor a pak importovány do aplikace. Znovupoužití v Tag-based systému je hlavně vyřešeno použitím dynamickým nebo klient-server odkazováním. Systém vytvoří základní grafiku obsahující skripty, které přepínají tagy v běhu (run-time). A protože je struktura aplikace plochá, uživatel musí změnit každý tag v systému a analyzovat efekt ve zbytku celé aplikace.

Průběh vývoje

Z počátku PC-based HMI a SCADA softwaru, uživatelé sestavovali grafiku a spojili ji s tagy, které reprezentují adresy v PLC nebo řídicího systému. Kroky uvedené níže, reprezentují typický vývoj tradičních Tag-based SCADA aplikací:

1. Nová HMI aplikace je vytvořena na jediném počítači.
2. Pro aplikaci jsou vytvořeny okna nebo obrazovky.
3. Pro okna je vytvořená grafika.

4. Definice tagů jsou importovány z PLC nebo manuálně konfigurovány.
5. Detekční alarmy a událostní skripty jsou definovány pro každý tag.
6. Tagy jsou napojeny na grafické elementy.
7. Jsou vytvořeny skripty grafických animací.
8. I/O tagy jsou definovány a spojeny do aplikací.
9. Jestliže je aplikace umístěna v klient-server prostředí, je znovu sestavena kvůli centralizování alarmů, detekci událostí, historickému archivování, grafiky a I/O serverům.
10. Změny v systému vyžadují vypnutí aplikace.

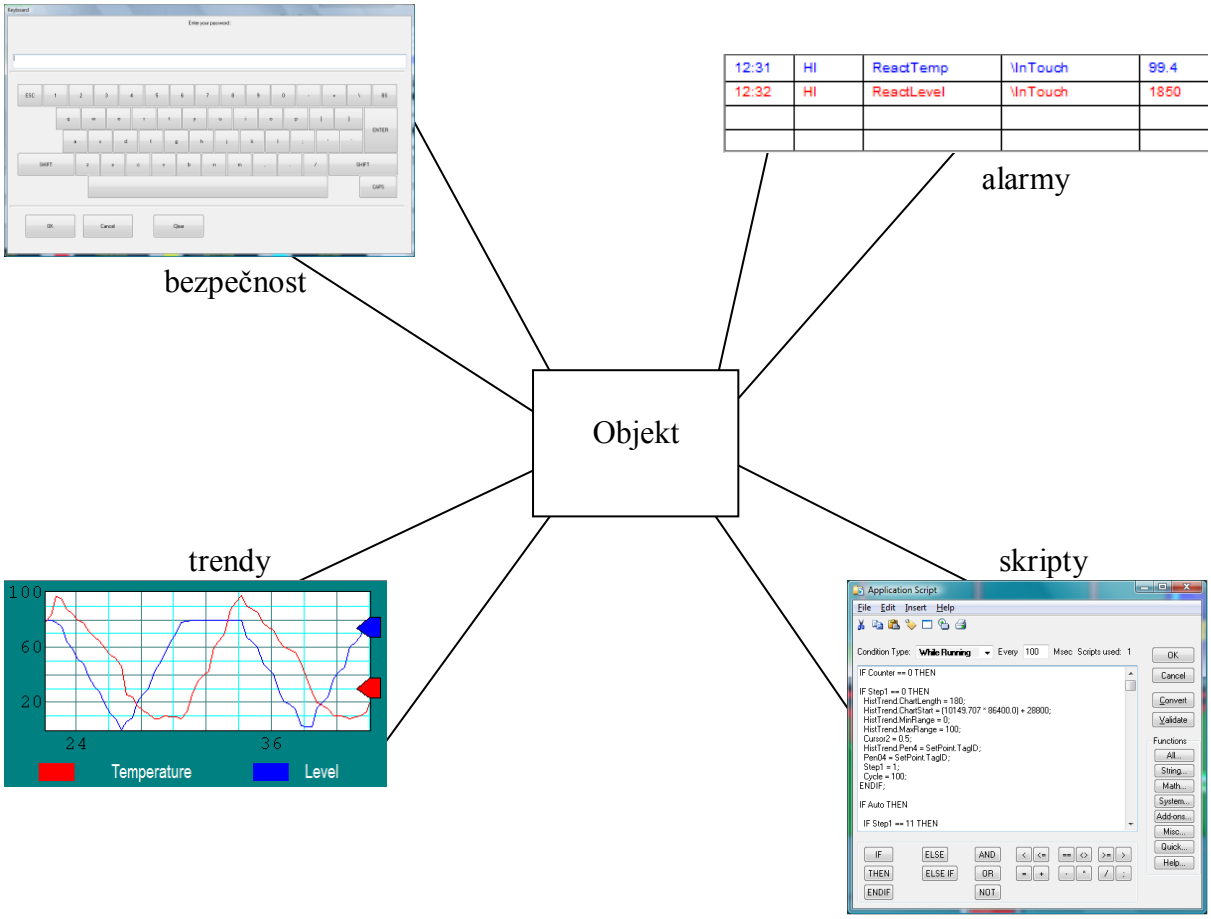
[GARBRECHT, 2006]

2.3 Component Object-based systems

Koncept Component-based je původně z oblasti informačních technologií. Jeho cílem je zajistit nástroje, které mohou uvolnit vývojáře od programových úloh, zatímco ve stejné době maximalizuje znovupoužití díky vývoji základních komponent. Tento přístup, nabízený Wonderware Industrial Application server a jeho podvrstvou softwarové architektury ArchestrA, je navrhován pro průmyslové zákazníky, kteří vyvíjejí, řídí a udržují supervizní řízení.

V aplikacích Component object-based SCADA, aplikační objekty obsahují aspekty nebo parametry připojené k zařízení, které reprezentují. Například, objekt Ventil může obsahovat všechny události, alarmy, bezpečnost, komunikaci a skriptování připojené k zařízení, viz obrázek 2.1.

Objekty nereprezentují pouze Plant vybavení. Mohou také modelovat základní kalkulace, metody přístupu k databázi, klíčové ukazatele výkonnosti (KPI), události monitorování stavu, operace přenosu dat v ERP a další produkční a výkonové úlohy. A protože tyto operace jsou standardní, je lehké je přidat do jedné nebo i více částí aplikace. Například, jestliže existuje standardní způsob údržby pumpy, můžeme zapouzdřit tuto funkci jako objekt a použít s jakoukoliv pumpou v aplikaci.



Obr. 2.1 – Ukázka vlastností objektu

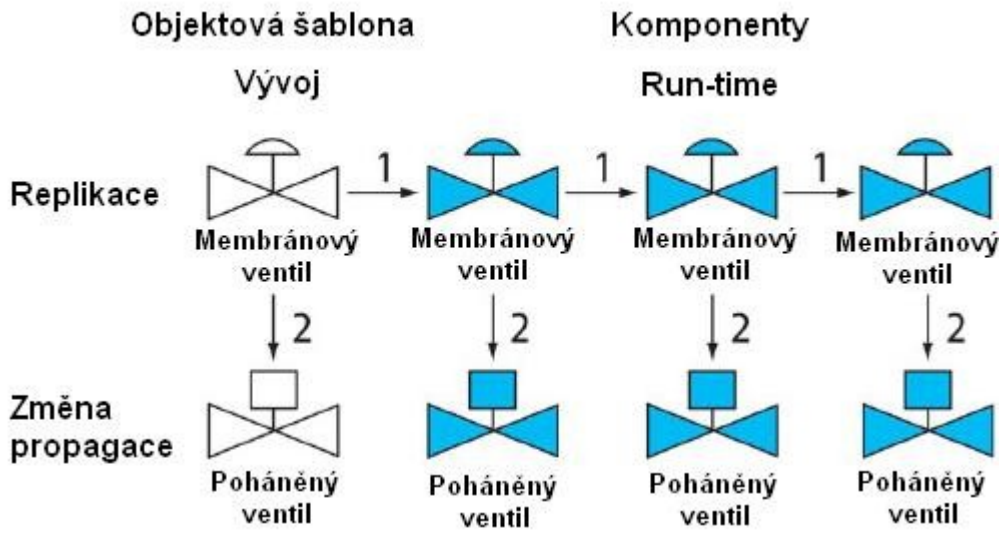
Výhodou je také to, že uživatelé mohou vyvíjet objekty jednou a potom je znovu použít v aplikaci, stejně jako v budoucích projektech.

Výroba aplikací má obvykle množství základních komponentů, zahrnujících typy:

- Zařízení a výbava typu Plant
- Operační procedury
- Procesní měření
- Kalkulace a grafické zobrazování

Vývoj Component object-based ulehčuje cookie-cutter přístup, v kterém malé softwarové programy mohou být vyvíjeny jako moduly objekty/kódy, přidané dohromady k utvoření aplikace. Kde je Component object-based SCADA systém odlišný je to, že pokud jsou

cookies odznačené, můžeme změnit značení a všechny cookies, které jsou už vyrobeny, jsou automaticky změněny.



Obr. 2.2 – Replikace objektů (Replication) a změna propagace (Change Propagation)

Na obrázku 2.2 je v horní řadě ukázána replikace objektu reprezentující membránový ventil. Replikace je procedura, během které jsou komponenty vytvořeny z objektových šablon. Další řada ukazuje změnu (z manuální do poháněné) stávající se propagovanou přes všechny ventily. Změna propagace je udělaná změnou objektových šablon (Object Template), které mohou způsobit některým nebo všem z asociovaných komponent změnu.

Pokud je SCADA balíček založen v Component object-based systému, je možná jak replikace, tak změna propagace a tím pádem zde existuje vztah rodič-dítě. Rodič, který je šablonou, je vyvinut jako první, a všechny ostatní komponenty jsou z něj replikovány nebo odvozeny. Tyto objekty, nazývající se "děti", jsou se svým rodičem zpětně svázány, takže změna v rodiči se projeví i v jeho dětech. Touto možností vznikají následující výhody:

- Aplikační tvorba je optimalizována použitím objektových šablon k automatickému generování komponent (replikace).
- Změny v projektech jsou lehce přizpůsobovány děláním změn v šabloně objektu a zdědění změn komponent přes změnu propagace.
- Další systémové změny a rozšíření jsou více cenově příznivější díky automatizované replikaci a změně propagaci.

Průběh vývoje

Po té, co jsou Plant modely zajištěny, je jednoduché implementovat funkce supervizního řízení. Deset kroků k vytvoření supervizní aplikace použitím průmyslového aplikačního serveru tedy jsou:

1. Stanovištní průzkum vede k porozumění návrhu výrobních operací nebo procesů.
2. Je vytvořen seznam podobných částí. Také jsou identifikovány zřetelné oblasti operací.
3. Objektové šablony jsou konfigurovány pro každé základní zařízení nebo komponentu. Tento proces upravuje standardy pro supervizní aplikaci a pro další aplikace, které budou vytvořeny v budoucnu.
4. Objektové šablony zařízení mohou být obsaženy v každé další šabloně k vytvoření více komplikovaných zařízení.
5. Objektové šablony zařízení mají atributy, které reprezentují reálné I/O proměnné dostupné v PLC nebo v řídicím systému. Tyto atributy jsou potom spojeny do I/O přes objekty pro integraci zařízení.
6. Aplikace může být sestavena použitím jednoduchého drag-and-drop obsaženého uvnitř IDE.
7. Komponenty jsou přiřazeny k bezpečnostním skupinám.
8. Model vytvořený v IDE může být nyní rozmístěný do počítačů, kde bude umístěna aplikace.
9. Grafika je konfigurována použitím softwaru HMI InTouch od společnosti Wonderware.
10. Jakmile je aplikace vyvinutá, systémová údržba je jednoduchá. Změny v objektových šablonách mohou být propagovány komponentám děti.

[GARBRECHT, 2006]

2.4 Porovnání obou systémů

Následující tabulka 2.1 ukazuje rozdíly mezi Component object-based a Tag-based architekturou. Můžeme zde vidět rozdíly obou přístupů ve vývoji a běhu. Další porovnání, porovnání úspor, lze vidět v kapitole 2.5. [GARBRECHT, 2006]

Tab. 2.1 – Rozdíly mezi Component object-based a Tag-based systémy

	Component object-based architektura		**Tag-based architektura**	
	Vývoj	**Běh**	**Vývoj**	**Běh**
Struktura aplikace	Hierarchická, objekty jsou tvořeny pomocí objektově orientované metodologie.	Hierarchický, komponenty reprezentují reálné přístroje a mohou se koordinovat s ostatními komponentami na jiných počítačích.	Hierarchický, grafický kontext je někdy vytvořen objektově orientovaným způsobem.	Plochý, monolitické instance softwaru běží na jednom nebo více strojích jako samostatné aplikace.
Grafický vývoj	Provádí se nakonec	N/A	Provádí se nejdříve	N/A
Skriptování	Vytvořeno v objektových šablonách jako část komponent.	N/A	Vytvořeno zvlášť, napojeny na grafickou reprezentaci.	N/A
Standardy	Striktně vynucené	N/A	Nejsou striktně vynucené	N/A
Globální aplikační změny	Propagovány z objektových šablon	Komponenty mohou být distribuovány, vyměněny nebo rozšířeny	Založeno na grafické nebo změněno pomocí tabulkových procesorů.	Požaduje rekompilaci aplikace.
Jak jsou reprezentována data	Logické konstrukce jako jsou fyzické zařízení (trupky, pumpy) nebo logické zařízení (PID smyčky, kalkulace) jsou reprezentovány jako objekty a komponenty.	N/A	Grafické zařízení jsou reprezentovány jako objekty nebo tagy.	N/A

2.5 Úspory

Úspory za životnost přidružené k jakýmkoliv SCADA vývojovým nástrojům mohou být kategorizovány do čtyř základních oblastí, jak je ukázáno v tabulce 2.2.

Tab. 2.2 – Čtyři oblasti úspor a jejich vysvětlení

Oblast úspor	Vysvětlení
Úspory počátečního vývoje související s generací aplikace.	Reprezentuje úspory, které vyústí z úspory času, když uživatelé vyvíjí aplikace definováním objektových šablon jednou, a potom generováním komponent z těchto šablon několikrát.
Úspory počátečního vývoje související s aplikačními změnami.	Reprezentuje úspory vývoje získané díky schopnosti změny propagace z objektových šablon ke všem komponentám odvozených z těchto šablon. Když je potřeba několik změn v aplikacích během vývoje, úspory se dají snadno spočítat.
Úspory údržby díky systémové životnosti.	Použitím distribuovaného systému se značně snižuje cena údržby přes možnost vzdáleně monitorovat, změnit a rozmístit software všem HMI uzlům v síti. Toto je zvláště důležité pro geograficky distribuované sítě, protože uživatelé mohou ušetřit čas i peníze eliminováním potřeby cestovat na každé místo pro údržbu nebo aktualizaci.
Úspory přes všechny oblasti.	Tyto úspory mají za následek znovupoužití šablon a aplikací vytvořených pro jeden projekt na dalších projektech. Společnosti tohle používají k řízení standardů v jejich projektech. Je to částečně prospěšné pro systémové integrátory, VARs, OEMs, stavitele strojů a operátory.

Jak už bylo na začátku řečeno, úspory mohou dosáhnout až 80 procent. Zde se jedná o časové úspory, které jsou samozřejmě pevně spjaty s finančními úsporami. Následující tabulka 2.3 ukazuje úspory mezi oběma přístupy Tag-based a Component object-based v příkladu ventilu s 27 instancemi a 6 I/O na každou instanci.

V Tag-based tedy potřebujeme 162 tagů (27 * 6) zatímco v Component object-based potřebujeme pouze jednu šablonu. Ovšem tato šablona obsahuje veškeré skriptování, alarmy, bezpečnost a další.

Tab. 2.3 – Úspory Tag-Based vs Component object-based při vytváření aplikace

Tag-based	Component object-based	Úspory
162 tagů * 0,4h každý = 64,8h	(2h * 1 šablona objektu) + (27 instancí * 0,4h každou) = 12,8h	52h nebo 80 procent

Tabulka 2.3 nám ukázala, že úspory při výrobě aplikace použitím Component object-based budou až 80 procent. Úspory při editaci aplikace jdou vidět v tabulce 2.4.

Tab. 2.4 – Úspory Tag-based vs Component object-based při editaci aplikace

Tag-based	Component object-based	Úspory
64,8h * 10 procentní změna = 6.48h	2h na každou objektovou šablonu * 10 procentní změny = 0,2h	6,28h nebo 96 procent

Tabulka 2.4 nám ukázala, že hlavní silou Component object-based jsou hlavně následné editace aplikací, které jsou nesrovnatelně rychlejší a tím pádem také levnější.

[GARBRECHT, 2006]

3 Vybrané vývojové a runtime softwarové prostředí

Ke zpracování mé aplikace jsem se rozhodl využít prostředí od firmy Wonderware, InTouch. Tedy Tag-based architekturu z důvodu její jednoduchosti a menšího rozsahu programované aplikace, která by nevyužila výhody poskytované Component object-based architekturou.

3.1 InTouch

InTouch je objektově orientované grafické prostředí od firmy Wonderware, kterou v České republice zastupuje firma Pantek. Je určené pro supervizní řízení, vizualizaci a sběru dat z technologických procesů SCADA/MMI. Umí graficky zobrazovat procesy, ovládat je nebo i animovat a to v reálném čase, jak lze vidět na obrázku 2.1. InTouch se vyvíjí už od roku 1987 rovnou pro MS Windows platformu.

Obr. 3.1 – Ukázka runtime okna programu InTouch [*Pantek*, 2009]

Pro sběr dat z technologických procesů je zařízena rozsáhlá nabídka komunikačních I/O serverů od Wonderware nebo třetích stran. Je zařízena také DDE nebo OPC komunikace. Kromě nástrojů pro grafické zpracování aktuálních stavů provozovaných technologií je součástí InTouch také zpráva distribuovaných historických dat i spolupráce s historizačními

databázemi Wonderware Historian Server, zprávu distribuovaných alarmů, které lze ukládat do databází typu MS SQL Server.

Obr. 3.2 – Ukázka technologického procesu InTouch [*Pantek*, 2009]

Součástí programu InTouch je také událostně orientovaný skriptovací jazyk s množstvím funkcí. Podporuje také různé technologické standardy jako je například, ActiveX, .NET nebo komunikace s relačními databázemi ADO/ODBC. Díky rozšiřujícím modulům Recipe Manager, SQL Access, SPC a sadě rozšiřujících nástrojů zahrnující také knihovny objektů s rozmanitou funkčností, díky nimž je vývoj aplikací velmi snadný. InTouch je také vybaven systémem InTouch Runtime Read-only, pomocí kterého je možné pouze prohlížet provozované použité procesy bez možnosti přímého zásahu do řízené technologie.

[VLACH, 1999],[*Pantek*, 2009]

3.2 Nástroje prostředí InTouch

Prostředí InTouch obsahuje velké množství nástrojů pro grafické kreslení, od klasických geometrických tvarů až po pokročilé, jako jsou tlačítka. Ke všem jdou přiřadit akce nebo je přímo animovat v závislosti na určité proměnné. Velkou částí, která se hojně využívá, jsou také Smart symboly, wizzardy a Symbol factory.

Smart symbol

Smart symboly se poprvé objevily v InTouch verzi 9.0 a jejich hlavním úkolem je využití Component object-based architektury. Tedy s dědičností, s cílem snížení počtu opakovaných operací. V podstatě se staly předchůdcem IDE Template v Component Object-based architektuře.

Smart symboly využívají takzvaných šablon, tedy složenin z několika základních grafických prvků spolu s předdefinovanými funkcemi jako například animace, skripty a jiné. Tyto šablony se uloží jako Smart symbol a do aplikace se zařazují už jako instance (pracovní funkční kopie). Pokud potom budeme chtít provést hromadnou úpravu, stačí upravit mateřskou šablonu Smart symbolu. Velká výhoda Smart symbolů je právě v jejich inteligentních úpravách a snadné propagaci i na vzdálených počítačích.

Smart symboly se řadí v knihovně, která umožňuje jednoduchou správu a přenositelnost (import a export).

Wizzard

Wizzard neboli „kouzelník“ je knihovna v prostředí InTouch, která obsahuje velké množství už připravených objektů s mnoha funkcemi. Jsou to například, hodiny, přepínače, různé typy světel nebo i postníky a SPC nástroje.

Knihovna je přehledně rozložena podle skupin, ze kterých se dá snadno vybrat potřebný prvek a umístit na pracovní plochu prostředí InTouch. Tyto symboly jdou omezeně upravovat, ale jelikož to jsou v podstatě pouze složeniny, lze je upravit rozbitím, úpravou a následným dalším složením.

Symbol factory

Jedná se o část Wizzardu, složku, která obsahuje velmi velký počet různých grafických objektů zakomponovaných do mnoha kategorií. Obsahuje hlavně obrázky různých technických zařízení (pumpy, potrubí, počítače a mnoho dalších), které se dají jednoduše upravovat a použít jako příkladný představitel reálného zařízení.

[TOMANEK, 2006]

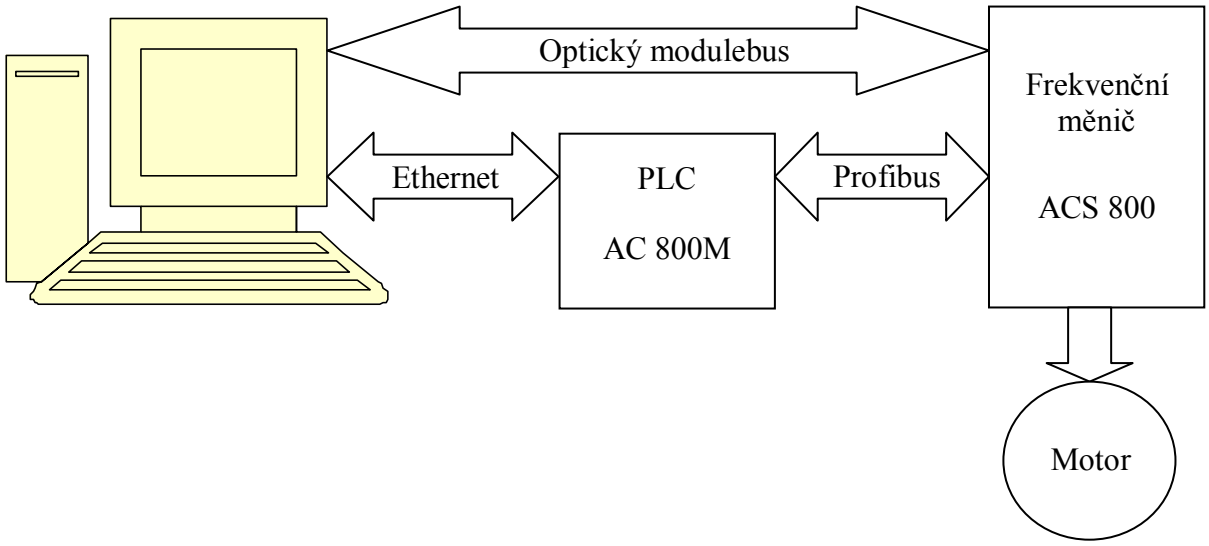
4 Laboratorní model

Laboratorním modelem je v této práci nízkovýkonový hliníkový asynchronní motor M2AA 112M s frekvenčním měničem ACS 800, PLC AC 800M a počítač. Model je umístěn na učebně F204.

4.1 Hardwarové propojení

Asynchronní motor M2AA 112M je napojen na frekvenční měnič ACS 800 a pomocí sběrnice Profibus spojen s PLC AC 800M od firmy ABB. Toto PLC je pak napojeno na počítač pomocí sběrnice Ethernet.

Hardwarové propojení mezi komponentami bylo již dříve realizováno a schéma propojení lze vidět na obrázku 4.1. Na obrázku 4.1 lze také vidět, že je možné také přímé propojení frekvenčního měniče rovnou s počítačem a to pomocí optického modulebus a programu DriveWindow. Toto spojení jsem ale nepoužíval, pouze prostřednictvím PLC.



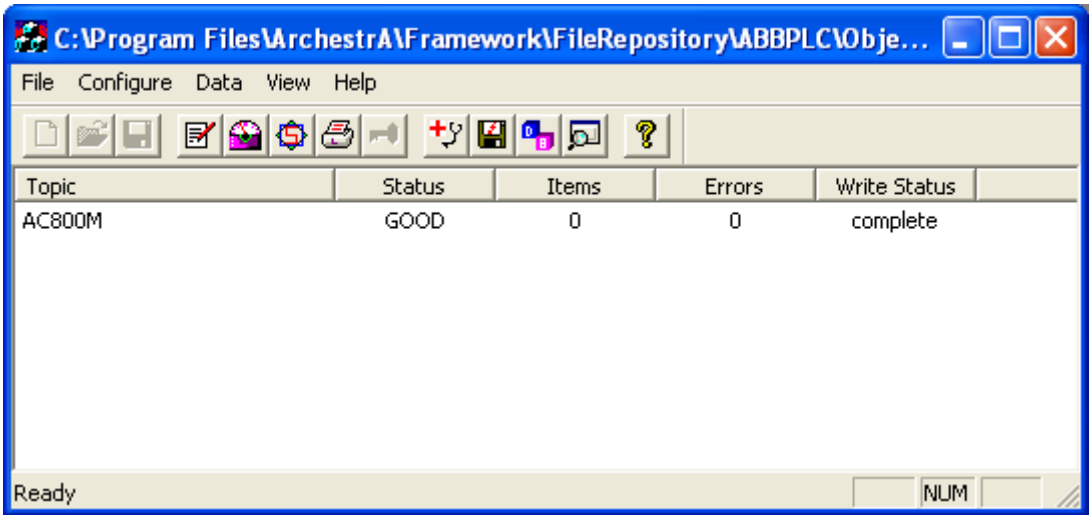
Obr. 4.1 – Schéma zapojení

4.2 Softwarové propojení

Softwarové propojení je zajišťováno pomocí OPC spojení. Bohužel program pro vizualizaci, který jsem použil na sestavení vizualizační aplikace, InTouch, neumí pracovat

s OPC, je třeba použít další program jako převodník. Tímto programem je OPC Link, který se napojí na OPC server a tak umožní komunikaci s PLC.

Pro správnou funkci programu OPC Link je třeba, aby všechny I/O proměnné (Items) nevykazovaly chyby (Errors). Pokud jsou u všech proměnných tyto chyby přítomné, je třeba překontrolovat, jestli je nahraný správný program v PLC protože má práce není jediná, která spolupracuje s tímto PLC. Tudíž se musí nahrát správný program, který komunikuje s frekvenčním měničem (Tento program byl už dříve realizován). Program se nahraje na PLC pomocí aplikace PLC Control Buldier AC 800M. Je to nástroj pro vytváření programů do PLC a také pro jejich nahrávaní do PLC. Tento program je určen přímo pro použité PLC a je dodávám přímo firmou ABB. [ZAVADIL, 2010]

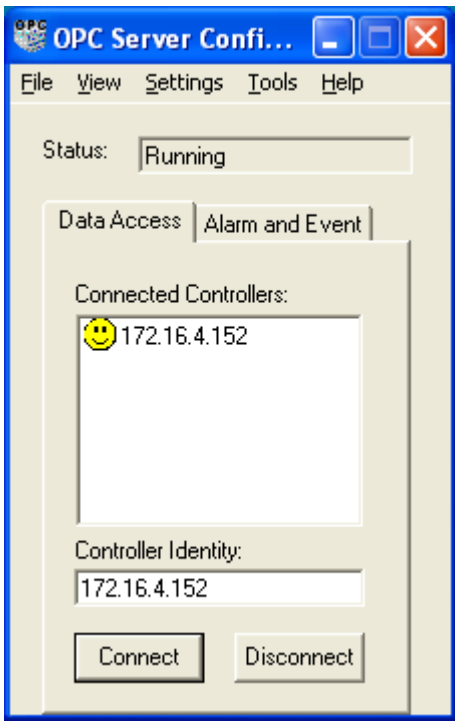


Obr. 4.2 – Okno programu OPC Link

Jak bylo v předchozích odstavcích řečeno, OPC Link se napojí na OPC server. Tento server je ve své podstatě program OPC Server, který po zadání IP adresy PLC, vytvoří komunikační most mezi aplikací (InTouch v zastoupení OPC Link) a PLC. Pokud je spojení úspěšné, zobrazí se potvrzující ikona (žlutý smajlík). V opačném případě se zobrazí ikona jiná (bílý kříž v červeném poli).

Vzhledem k softwarové komunikaci je ještě nutné zdůraznit, že program v PLC nepracuje rovnou s hodnotami zadanými v prostředí InTouch nebo přeposlanými pomocí OPC Link ale vnitřně je přepočítává na další hodnoty. Je to kvůli tomu, že frekvenční měnič neumí pracovat s diskrétními hodnotami (True, False) nebo přímými hodnotami otáček, ale pracuje

na základu řídících slov. Například, funkce Připrav, která připravuje měnič k funkci je řídící slovo 1030, ale v prostředí InTouch se jedná o diskrétní hodnotu.



Obr. 4.3 – Okno programu OPC Server

Veškeré proměnné pro komunikaci vizualizačního prostředí InTouch s programem v PLC jsou k dispozici v programu OPC Link, který byl pro toto už dříve nakonfigurován. Pro správnou komunikaci je pak ještě nutné zadat před proměnnou ve vizualizačním prostředí InTouch písmeno „r“ nebo „d“ jestli se jedná o diskrétní nebo reálnou hodnotu. Ovšem pouze v položce Item, která je napojení na OPC Link. Klasické názvy proměnné v prostředí InTouch mohou být jakékoliv, jedná se totiž pouze o vnitřní proměnnou samotného vizualizačního prostředí InTouch.

5 SPC nástroje

SPC neboli Statistical Process Control (Metody pro řízení kvality) jsou metody pro monitorování určitých procesů pomocí regulačních diagramů nebo i jiných funkcí. Měřená data se ukládají do databáze, ke které přistupují SPC nástroje a využívají je k vizualizaci, následně také pro řízení kvality výrobního procesu. [STANÍČEK, 2009]

SPC nástroje, které jsem vybral pro pozdější přidání do vizualizační aplikace, jsou:

- Regulační diagram
- Histogram

Metody řízení kvality můžeme rozdělit do tří rozsáhlých skupin:

- **Pochopení procesu** – Proces je detailně zmapován a sledován pomocí regulačních diagramů (Control chart), které se používají k identifikaci odchylek.
- **Pochopení příčin odchylek** – Jakmile je pomocí regulačního diagramu zjištěna odchylka či nedostatek, můžeme použít například Paretův diagram k oddělení podstatných faktorů v procesu zabezpečení jakosti.
- **Eliminace zdrojů příčiny odchylek** – Dále je zaměřeno úsilí na odstranění příčin. Obecně to zahrnuje klasickou práci, kdy se odstraní chyba, případně se provede školení.

Dohromady existuje sedm základních statistických nástrojů pro vizualizaci a řízení procesů:

1. Diagramy Ishikava – vyhledávání příčin
2. Časové diagramy – sledování trendů
3. Bodové diagramy – měření závislostí
4. Vývojové diagramy – zobrazení procesů
5. Paretovy diagramy – zaměření na klíčové problémy
6. Histogram – statistické chování procesů
7. Regulační diagramy (Control chart) – řízení procesů

Všechny tyto nástroje pomáhají k lepšímu porozumění procesů a zajišťování vyšší jakosti a tedy i efektivity výroby. [MILITKÝ, KŘEMENÁKOVÁ, 2000]

5.1 Regulační diagram (Control chart)

Regulační diagram neboli Control chart je hlavním nástrojem statistické regulace procesů. Vytvořil jej Američan W. A. Shewhartem v roce 1926. Jedná se o grafickou pomůcku, která odděluje náhodné příčiny od identifikovatelných příčin. Ty pomáhá oddělit.

Když se vytváří regulační diagram, musíme vyjít z principu Centrální limitní věty. Ta se považuje za základní tvrzení statistiky a zahrnuje v sobě tyto aspekty:

1. Když odebíráme jednotky, u kterých chceme zjistit regulovanou veličinu neboli znak jakosti, po skupinách (tzv. podskupinách) a hodnoty tohoto znaku jakosti v podskupinách zprůměrujeme a rozdělení průměrů aproximujeme k normálnímu rozdělení tím více, čím jsou tyto podskupiny větší. Bylo prokázáno, že stačí podskupiny o velikosti pět jednotek, aby toto bylo splněno.
2. Střední hodnota rozdělení průměrů je totožná jako střední hodnota jednotlivých hodnot.
3. Směrodatná odchylka rozdělení průměrů je $\sqrt{n}$ krát menší než směrodatná odchylka jednotlivých hodnot.

Princip regulačních diagramů je vcelku jednoduchý, ale musí se držet určitých pravidel, která jsou velmi důležitá pro správné používání:

- Je třeba v pravidelných časových intervalech provádět náhodný odběr předustanoveného počtu členů, které tvoří podskupiny.
- U produktů stejného druhu vyrobených za stejných podmínek musíme zjistit stejný znak jakosti jako je třeba určitý rozměr.
- Musíme vypočítat pro každou podskupinu jednu nebo více výběrových charakteristik jako jsou průměr, rozpětí a směrodatná odchylka.
- Zjištěné hodnoty se musí zakreslit do regulačního diagramu.
- U regulačního diagramu se musí provést analýza.

Analýzou regulačního diagramu rozumíme zjišťování stavu kdy je proces “pod kontrolou“ nebo “mimo kontrolu“. Proces je mimo kontrolu, pokud body nebo bod leží mimo regulační meze, případně pokud body vykazují nenáhodná uskupení nebo trendy. Pokud tato situace nastane, je potřeba provést analýzu procesu, vyhledat a odstranit příčinu.

Ovšem mohou nastavit dva druhy chyb. První chybou je, když je příslušný proces "pod kontrolou", ale některý z bodů leží mimo regulační meze. Na tomto základě můžeme špatně usuzovat, že je tento proces "mimo kontrolu" a vznikají zbytečné výdaje na zjištění příčiny. Tato chyba je 1. typu a značí se α (riziko zbytečného signálu). Druhá chyba, která může vzniknout, je kdy proces není "pod kontrolou", ale hodnoty charakteristiky leží uvnitř regulačních mezí. Pak je proces nesprávně posuzován jako statisticky zvládnutý a opět vznikají zbytečné náklady při selhání schopnosti vést k odhalení zvláštních příčin, které způsobují produkování neshodných jednotek. Pak tedy jde o chybu 2. typu, která se značí β (riziko chybějícího signálu).

Druhy statistické regulace

Podle povahy regulované veličiny dělíme statistickou regulaci na dva druhy:

1. Regulace měřením

Regulace měřením se používá v případě, kdy znak jakosti podle kterého proces regulujeme, je spojitou náhodnou veličinou. Například rozměry, vzdálenosti nebo vlhkost. Pro regulaci měřením stačí 4 až 5 jednotek v podskupině. Pro tuto regulaci je typické sestavování regulačních diagramů do dvojic. Charakteristiky popisující polohu procesu sestavujeme v prvním, v druhém analyzujeme variabilitu procesu. Nejčastěji se používají dvojice diagramů $(\bar{x}, R)$ a $(\bar{x}, s)$, tedy (průměr, rozpětí) a (průměr, směrodatná odchylka).

a) Regulační diagram pro samostatné hodnoty

Tento regulační diagram se používá hlavně v případech, kdy se data, která jsou pomocí tohoto regulačního diagramu zpracovávána, akumulují pomaleji.

Hodnotu střední přímky vypočteme ze vztahu:

$$CL = \bar{x} = \frac{\Sigma x}{n} \tag{5.1}$$

kde *x* je jednotlivá hodnota měření, n je počet použitých vzorků

Hodnotu horní a dolní regulační meze vypočteme ze vztahů:

$$UCL(LCL) = \bar{x} + (-)3\,.\,s \tag{5.2}$$

kde *s* je směrodatná odchylka.

$$s = \sqrt{\left(\frac{(\sum x^2 - n.\bar{x}^2)}{n-1}\right)} \quad (5.3)$$

kde *n* je počet použitých vzorků.

b) Regulační diagram pro (průměr, rozpětí)

Tento regulační diagram používá pro podskupiny malého rozsahu ($n \leq 8$) a minimální doporučeným rozsahem podskupin 4 až 5. Minimální počet podskupin je 20 až 25. Diagram pro průměr ($\bar{x}$) umožňuje analyzovat polohu procesu a její odchylky. Diagram pro rozpětí (R) nám umožňuje analyzovat stejnosměrnosti procesu. V porovnání s třetí metodou je získávání hodnot mnohem nákladnější a časově náročnější, ale naměřené číslové hodnoty mají větší vypovídající schopnost.

Hodnotu střední přímky pro diagram x vypočteme ze vztahu:

$$CL = \bar{\bar{x}} = \frac{\sum \bar{x}}{n} \quad (5.4)$$

kde *n* je počet použitých vzorků, $\bar{x}$ je průměr hodnot (viz rovnice č. 5.1).

Hodnotu střední přímky pro diagram R vypočteme ze vztahu:

$$CL = \bar{R} = \frac{\sum R}{n} \quad (5.5)$$

kde *R* je rozptyl hodnot v podskupině, *n* je počet použitých vzorků.

$$R = x_{max} - x_{min}$$

kde x_{max} je maximální a x_{min} minimální hodnota. (5.6)

Hodnotu horní a dolní regulační meze diagramu x získáme:

$$UCL\ (LCL) = \bar{\bar{x}} + (-)A_2\ .\ \bar{R} \quad (5.7)$$

kde A_2 je statistická konstanta UCL (LCL) diagramu ($\bar{x}, R$) pro *x*, $\bar{R}$ je průměr rozptylu podskupiny (viz rovnice č. 5.5).

Hodnotu horní a dolní regulační meze diagramu *R* získáme:

$$UCL\ (LCL) = D_4(D_3)\ .\ \bar{R} \quad (5.8)$$

kde D_4 a D_3 jsou Statistické konstanty UCL(LCL) diagramu $(\bar{x}, R)$ pro R, $\bar{R}$ je průměr rozptylu podskupiny (viz rovnice č. 5.5).

c) Regulační diagram pro (průměr, směrodatná odchylka)

Tyto regulační diagramy se používají hlavně u větších rozsahů podskupiny, hlavně tam kde se data vyhodnocují počítačem. Směrodatná odchylka je citlivější než rozpětí pro stanovení variability procesů.

Hodnotu střední přímky pro diagram *x* vypočteme ze vztahu:

$$CL = \bar{\bar{x}} = \frac{\sum \bar{x}}{n} \tag{5.9}$$

kde *n* je počet použitých vzorků, $\bar{x}$ je průměr hodnot (viz rovnice č. 5.1).

Hodnotu střední přímky pro diagram *s* vypočteme ze vztahu:

$$CL = \bar{s} = \frac{\sum s}{n} \tag{5.10}$$

kde *s* je směrodatná odchylka (viz rovnice č. 5.3), *n* je počet použitých vzorků.

Hodnotu horní a dolní regulační meze diagramu *x* získáme:

$$UCL\ (LCL) = \bar{\bar{x}} + (-)A_3\ .\ \bar{s} \tag{5.11}$$

kde $\bar{s}$ je průměr směrodatné odchylky podskupiny (viz rovnice č. 5.10), A_3 je statistická konstanta UCL (LCL) diagramu $(\bar{x}, s)$ pro *x*

Hodnotu horní a dolní regulační meze diagramu *s* získáme:

$$UCL\ (LCL) = B_4(B_3)\ .\ \bar{s} \tag{5.12}$$

kde $\bar{s}$ je průměr směrodatné odchylky podskupiny (viz rovnice č. 5.10), B_4 a B_3 jsou Statistické konstanty UCL(LCL) diagramu $(\bar{x}, s)$ pro *s*

[*Wonderware® FactorySuite™ SPCPro™*, 1999]

Přepočítací koeficienty

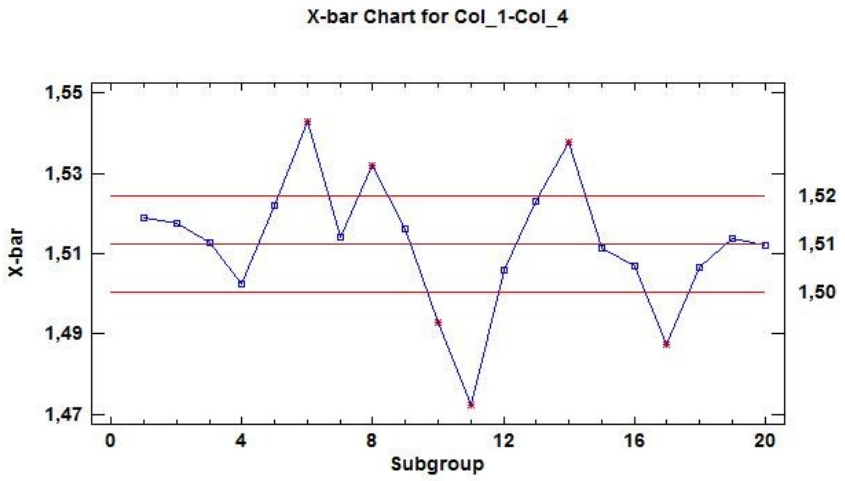
V tabulce 5.1 můžeme vidět seznam přepočítávacích koeficientů. První sloupec je rozsah výběru *n*, tedy počet podskupin. Druhý až sedmý sloupec jsou samotné přepočítávací koeficienty a tři poslední sloupce jsou koeficienty používané k jejich vyčíslení.

Tab. 5.1 – Tabulka přepočítávacích koeficientů

rozsah výběru	A_2	A_3	B_3	B_4	D_3	D_4
2	1,8806	2,6586	0	3,2664	0	3,2686
3	1,0231	1,9545	0	2,5684	0	2,5735
4	0,7289	1,6281	0	2,2662	0	2,2828
5	0,5763	1,4273	0	2,0889	0	2,1134
6	0,4833	1,2872	0,0300	1,9700	0	2,0039
7	0,4193	1,1819	0,1180	1,8820	0,0758	1,9242
8	0,3726	1,0991	0,1847	1,8153	0,1359	1,8641
9	0,3367	1,0317	0,2390	1,7610	0,2596	1,7404
10	0,3082	0,9753	0,2843	1,7157	0,2232	1,7768

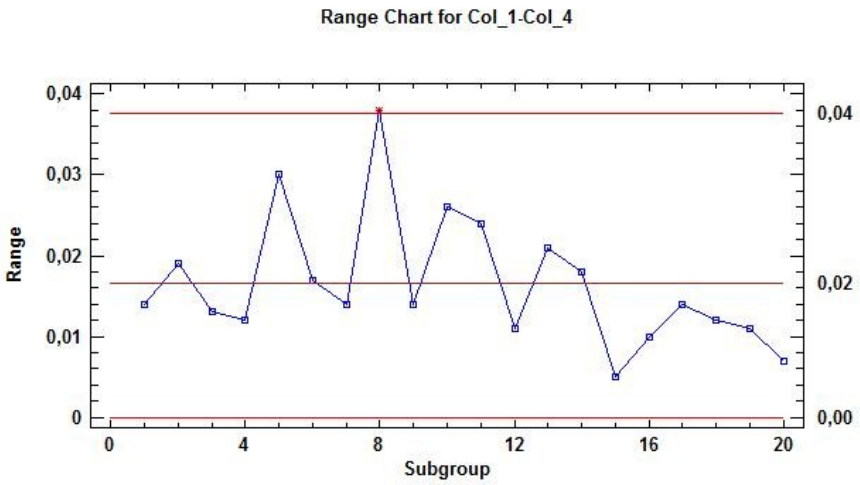
Kontrola správnosti dat

Abych měl jistotu, že jsem provedl správné výpočty pro Regulační diagram *(x,R)*, využil jsem program Statgraphics Centurion od firmy Statpoint Technologies ve verzi 16.1. Tento program obsahuje velké množství statistických výpočtů a mezi nimi i výpočty Regulačních diagramů. Data stačí jednoduše umístit do tabulkového rozhraní, podobného MS Excelu, označit použitá data a spustit potřebný výpočet.



Obr. 5.1 – Regulační diagram *x* v programu Statgraphics

Program poté vygeneruje grafy a balíček dalších dat. Porovnáním těchto grafů a dat jsem došel k názoru, že mé výpočty byly správné. Grafy lze vidět na obrázcích 5.1 a 5.2. S porovnáním na obrázky 5.3 a 5.4. [*Statgraphics Centurion*, ©2011]



Obr. 5.2 – Regulační diagram *R* v programu Statgraphics

2. Regulace srovnáváním

Pokud máme regulační veličinu jako diskrétní náhodnou veličinu (např. počet vad na odlitku), používáme regulaci srovnáváním. Tyto informace je možné získat mnohem levněji a rychleji než regulací měřením, ale získáme také menší množství informací o regulovaném procesu. Rozsah podskupin musí být také mnohem větší, ale také pracujeme pouze s jedním regulačním diagramem. Existují čtyři druhy regulačních diagramů:

a) Pro podíl neshodných jednotek v podskupině (regulační diagram pro p)

Tento regulační diagram je vhodný pro případ, kdy se rozsah podskupin mění. Rozsah podskupiny by se měl pohybovat od 50 do 200 členů. Počet podskupin by měl být minimálně 25.

b) Pro počet neshodných jednotek v podskupině stejného rozsahu n (regulační diagram pro np)

Také regulační diagram np je založeno na binomickém rozdělení a používá se, když máme konstantní velikosti podskupin.

c) Pro počet neshod v podskupině (regulační diagram pro c)

Tento regulační diagram je založen na Poissonově rozložení. Počet neshod v podskupině (odlitek nebo třeba celá součást) je výběrová charakteristika. Podskupiny musí být stejné velikosti a jako znak jakosti může být třeba počet vad na jednotlivých součástech.

d) Pro počet neshod (vad) na jednotku v podskupině (regulační diagram pro u)

Tento regulační diagram je určen na hodnocení podle průměrného počtu neshod neboli vad na objektech různé velikosti.

Fáze statistické regulace

Statistickou regulaci určitého procesu můžeme rozdělit do čtyř postupových fází, podle kterých se udržuje proces na stabilní úrovni.

1. Fáze přípravná

Před samotným prováděním analýzy procesu je třeba vybrat odpovědi na několik otázek, které nám dají první pohled:

- Jaký bude parametr procesu, tzn. regulovaná veličina?
- Jaká bude délka kontrolního intervalu?
- Jakým způsobem vytvoříme podskupiny?
- Jak budeme zjišťovat regulované veličiny?
- Jaký bude rozsah podskupin?
- Jaký použijeme regulační diagram?
- Jaká bude organizace sběru informací o procesu?

2. Fáze analýzy a zabezpečení statistické analýzy

Cílem pro tuto fázi je zajistit, aby byla variabilita procesu způsobena pouze náhodnými vlivy. Zde je potřeba identifikovat a odstranit zvláštní příčiny a zajistit aby se už nemohly opakovat.

3. Fáze analýzy a zabezpečení způsobilosti procesu

Zde zkoumáme, jestli je proces schopen dosáhnout potřebám požadovaných vlastností (například normy nebo technické prostředky). Ke způsobilosti procesu se pak využívá tzv. index způsobilosti *Cp*. Index způsobilosti ukazuje, jak jsou data centrovány mezi tolerančními mezemi.

$$Cp = \frac{USL - LSL}{6s} \tag{5.13}$$

kde USL je hodnota horní toleranční meze, LSL je hodnota dolní toleranční meze, *s* je směrodatná odchylka.

$$s = \sqrt{\frac{n \,.\sum x^2 - (\sum x)^2}{n \,.(n-1)}} \tag{5.14}$$

kde *n* je počet měření/vzorků.

Pokud má proces index způsobilosti aspoň ≥ 1,33, je považován za uspokojivý. Ovšem index způsobilosti ukazuje pouze přesnost procesu. Pro polohu procesu vzhledem k tolerančnímu poli se používá charakteristika *Cpk*, která ukazuje, jestli je nebo není zařízení seřízeno na střed tolerančního pole. Pokud proces není nastaven na střed tolerančního pole, může vykazovat vysoké hodnoty *Cp*, ale stejně může dojít k situaci, kdy se vyrábí zbytečně vysoké procento neshodných jednotek.

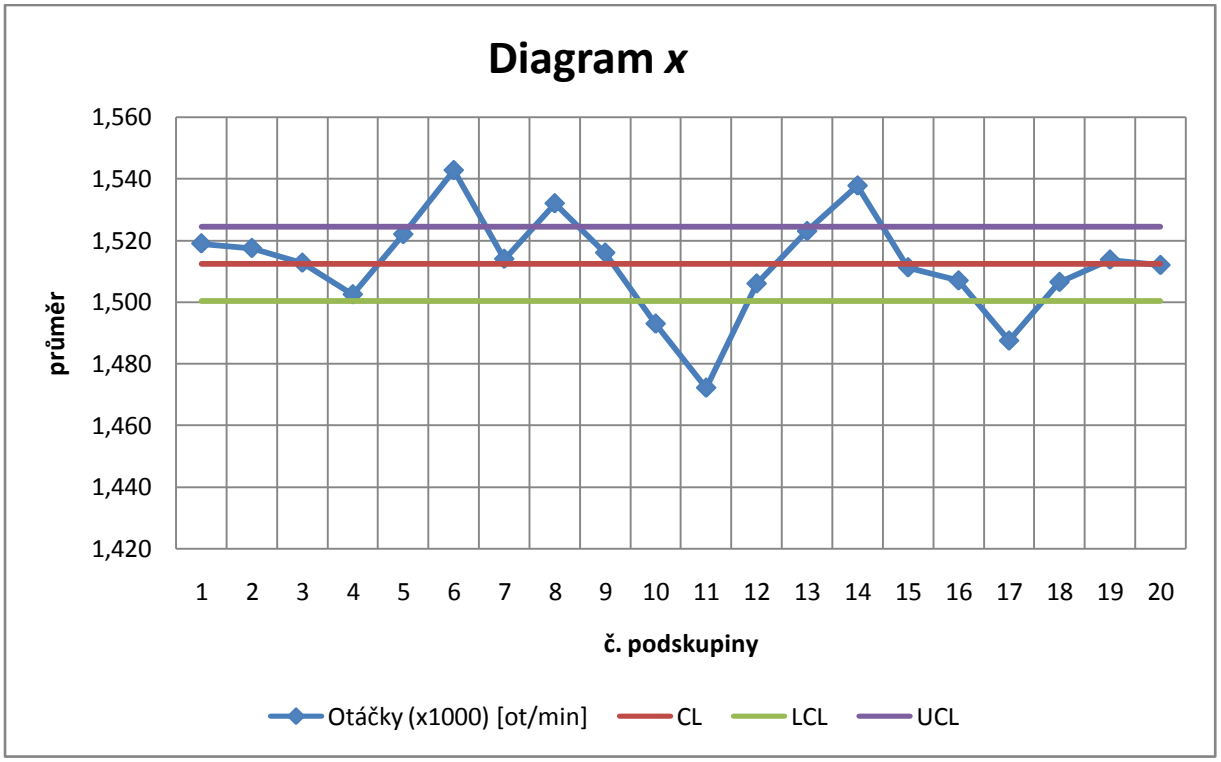
$$Cpk = min\left(\frac{USL - \bar{\bar{x}}}{3s}, \frac{\bar{\bar{x}} - LSL}{3s}\right) \tag{5.15}$$

kde USL je hodnota horní toleranční meze, LSL je hodnota dolní toleranční meze, $\bar{\bar{x}}$ je střední hodnota procesu, *s* je směrodatná odchylka.

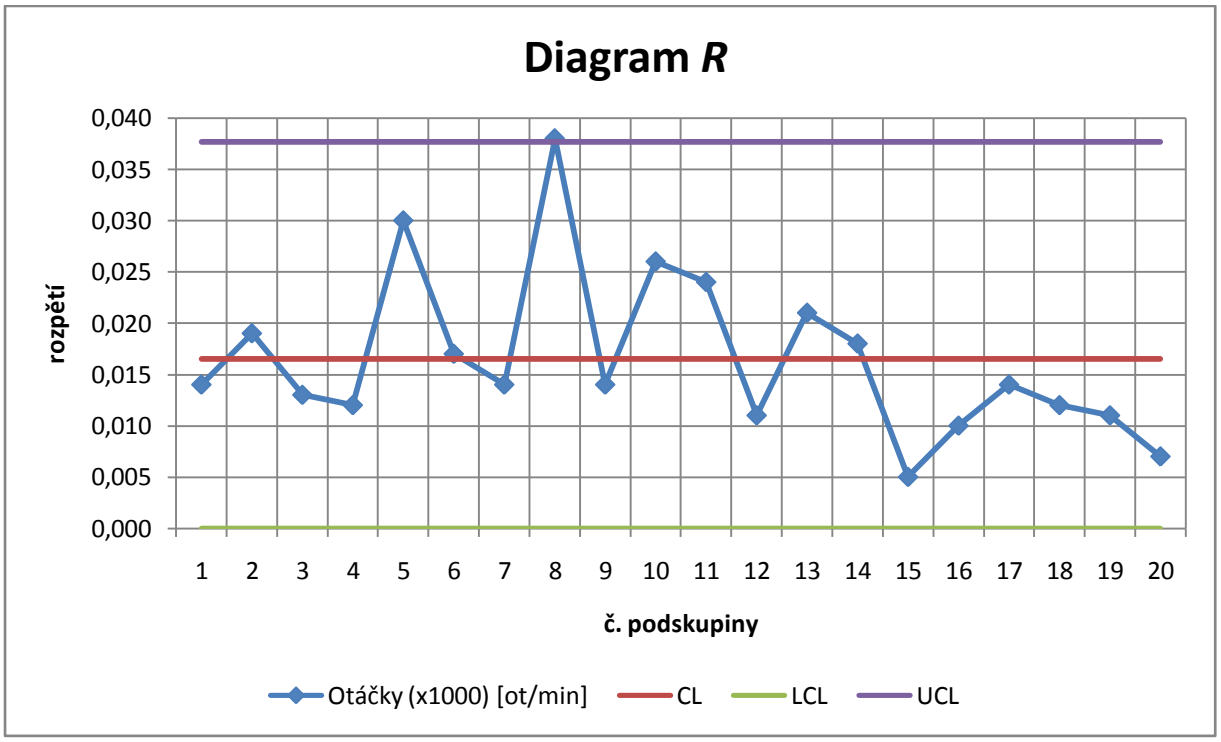
Hodnota *Cpk* by měla také dosahovat aspoň ≥ 1,33.

4. Fáze vlastní statistické regulace

Zde je cílem pomocí již zkonstruovaného regulačního diagramu zjistit a odstranit všechny poruchy ve stabilitě procesu. [NOSKIEVIČOVÁ, 1996]



Obr. 5.3 – Regulační diagram *x*



Obr. 5.4 – Regulační diagram *R*

5.2 Histogram

Histogram přestavuje grafické znázornění rozdělení četností. Může se jednat například o rozměry výrobků, chemické složení, pevnosti, napětí apod. nebo o hodnoty činitelů ovlivňující jakost jako je třeba rychlost, teplota, tlak, atd.

Histogram je sloupcový graf, jehož základna (osa x) obsahuje několik sloupců, které odpovídají šířce intervalu h. Osa y vyjadřuje četnost hodnot sledované veličiny. Díky histogramu můžeme mít velice rychlý přehled o statistickém souboru a jeho rozdělení, statistické charakteristiky atd. [POKORNÝ, KOZUB, 1998]

Postup při sestavování histogramu je následující:

1. Z naměřených hodnot se vypočte rozpětí R

$$R = x_{max} - x_{min}$$

kde x_{max} je maximální a x_{min} minimální hodnota. (5.16)

2. Vypočteme šířku vnitřního intervalu h – Tato hodnota musí být rozdělena do intervalů stejné velikosti, tak aby hodnota x_{min} ležela v prvním intervalu a hodnota x_{max} ležela v intervalu posledním. Samotnou šířku zjistíme tak, že rozpětí R vydělíme číslem 1, 2, 5 nebo jejich násobky abychom získali 7 až 20 dílčích intervalů o stejné šířce.
3. Stanovíme hranice intervalu. Minimální hodnota musí být v prvním a maximální v posledním intervalu. Dolní hranice prvního intervalu stanovíme tak, že musí platit $x_{D1} < x_{min}$ a současně musí být hraniční hodnota o jeden řád hodnotu přesnější než ostatní naměřené hodnoty. Tedy, pokud jsou naměřené hodnoty v desetinách, hraniční hodnota bude v setinách na číslici 5 (2,20**5** < 2,22). Horní hodnota se určí tak, že k hraniční hodnotě x_{D1} přičteme šířku intervalu h:

$$x_{H1} = x_{D1} + h \quad (5.17)$$

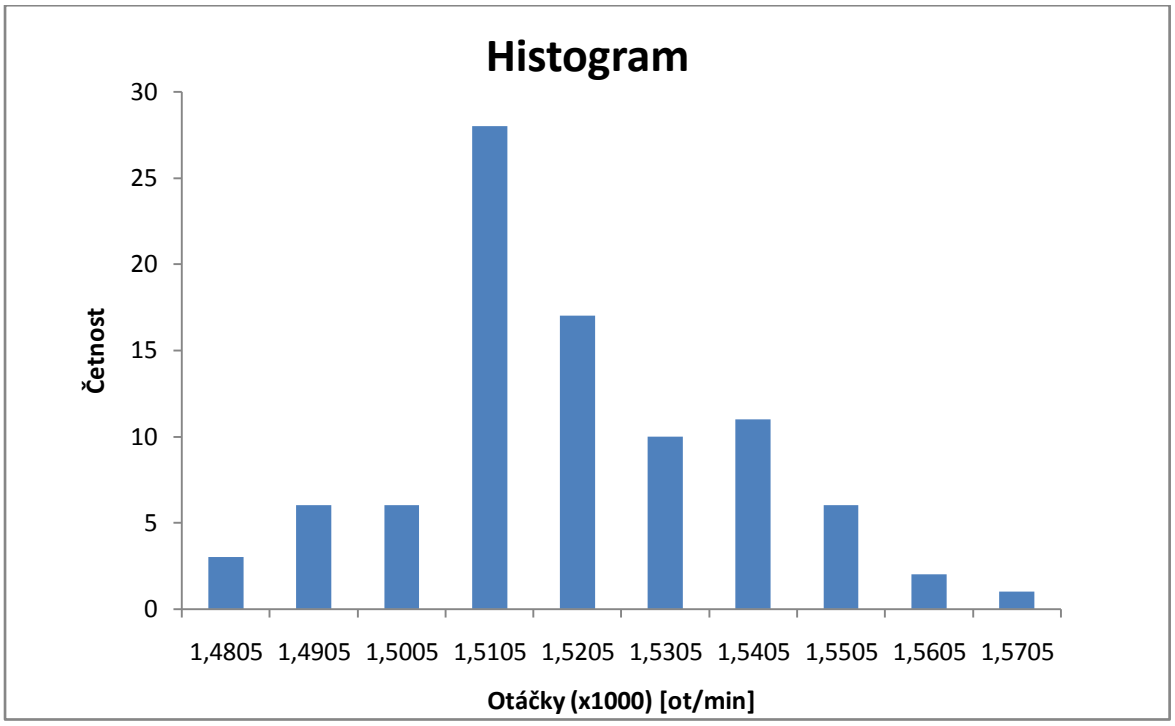
kde x_H je horní hranice určitého intervalu, x_{D1} je dolní hranice určitého intervalu, h je šířka intervalu.

Další hranice se určí podobně:

$$x_{D2} = x_{H1} \quad (5.18)$$

$$x_{H2} = x_{H1} + h \tag{5.19}$$

4. Sestavíme tabulku četností, a zaznačíme do ní naměřené hodnoty a v intervalech zaznamenáme četnost.
5. Nakonec sestavíme vlastní histogram. Na osu x naneseme hranice intervalů a na osu y naneseme hodnoty četností. Potom sestrojíme vlastní sloupcový graf.



Obr. 5.5 – Histogram

Analýza histogramu

Analýza histogram probíhá na základě jeho vzhledu. Tedy samotné rozdělení statistického souboru a tvaru grafu, který vypovídá hlavně o systematických vlivech, které působí na určitý parametr jakosti. Histogram může nabývat také různých tvarů:

- Normální (zvonovitý)
- Plochý
- Asymetrický
- Normální s izolovanými hodnotami
- Dvouvrcholový
- Hřebenový
- Useknutý

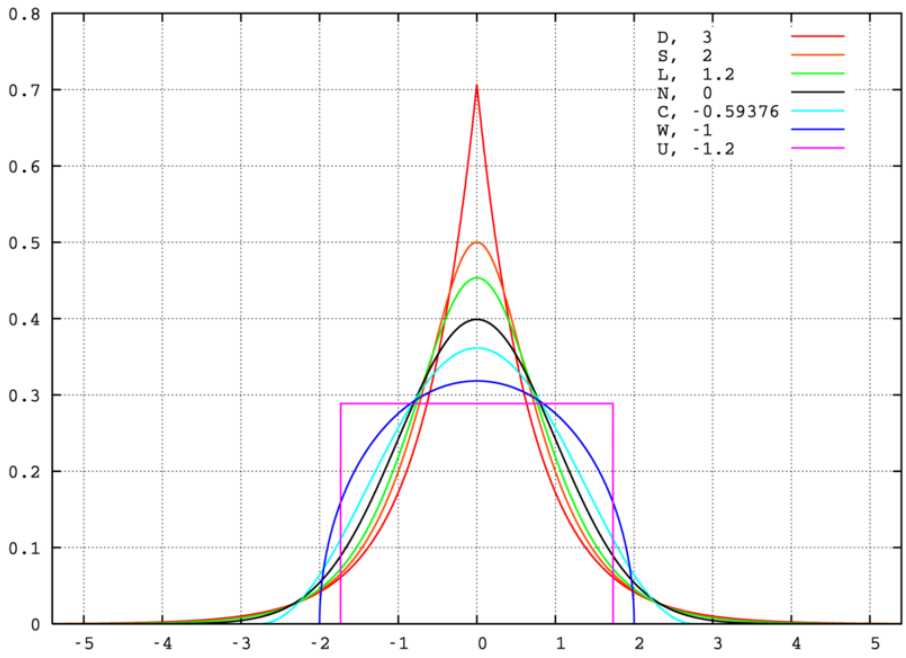
Různé typy odchylek od normálního (zvonovitého) tvaru histogramu nám mohou ukázat působení různých systematických vlivů. Tyto systematické vlivy je pak na základě této analýzy histogramu třeba odhalit a případně odstranit. [NOSKIEVIČOVÁ, 1996]

Dále se na Histogramu určuje šikmost a špičatost pro snazší popis:

Špičatost (Kurtosis) vyjadřuje míru rozložení distribuční funkce Gaussova rozdělení. Pro křivku s hodnotou špičatosti 3 je rozložení normální. Křivka s vysokým vrcholem nebo protažením má hodnotu vyšší než 3 a pro tlustější křivky má hodnotu menší než 3. Viz obrázek 5.6.

$$Kurtosis = \frac{m_4}{m_2^2} \tag{5.20}$$

kde m_2 a m_4 jsou druhý a čtvrtý moment přibližného průměru vzorku.



Obr. 5.6 – Ukázka špičatosti křivek [SWEEP, 2006]

Šikmost (Skew) je protažení křivky než normální rozložení na levou nebo pravou stranu. Pokud je křivka symetrická, je tedy šikmost nulová. Křivka, která má protažení směrem k velkým hodnotám má svoji šikmost pozitivní. Viz obrázek 5.7.

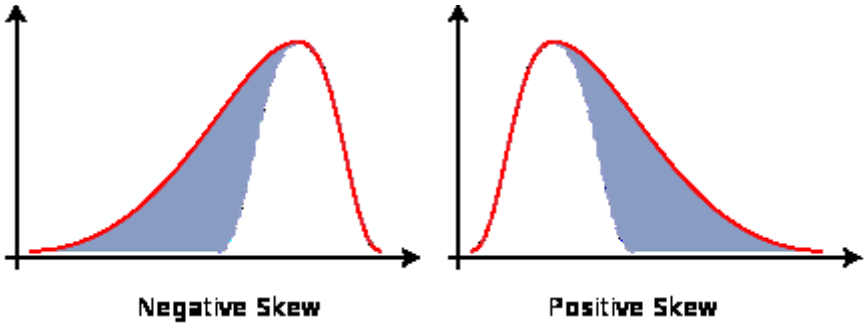
$$Skewness = \frac{m_3}{m_2^{3/2}} \tag{5.21}$$

kde m_2 a m_3 jsou druhý a třetí moment přibližného průměru vzorku.

Další udávané hodnoty v histogramech jsou Průměr, Horní a Dolní regulační mez. Průměr se počítá pomocí vzorce:

$$Mean = \frac{\sum x}{n \,.N} \tag{5.22}$$

kde *n* je počet měření/vzorků, *N* je počet zobrazených vzorků.



Obr. 5.7 – Ukázka šikmosti křivek [HERMANS, 2008]

Horní a dolní regulační meze se pak vypočítají:

$$UCL\ (LCL) = \ Mean + (-)3\ .\ s \tag{5.23}$$

kde *s* je směrodatná odchylka.

$$s = \sqrt{\frac{n_t \,.\sum x^2 - (\sum x)^2}{n_t \,.(n_t - 1)}} \tag{5.24}$$

$$n_t = n\ .\ N \tag{5.25}$$

kde *n* je počet měření/vzorků, *N* je počet zobrazených vzorků.

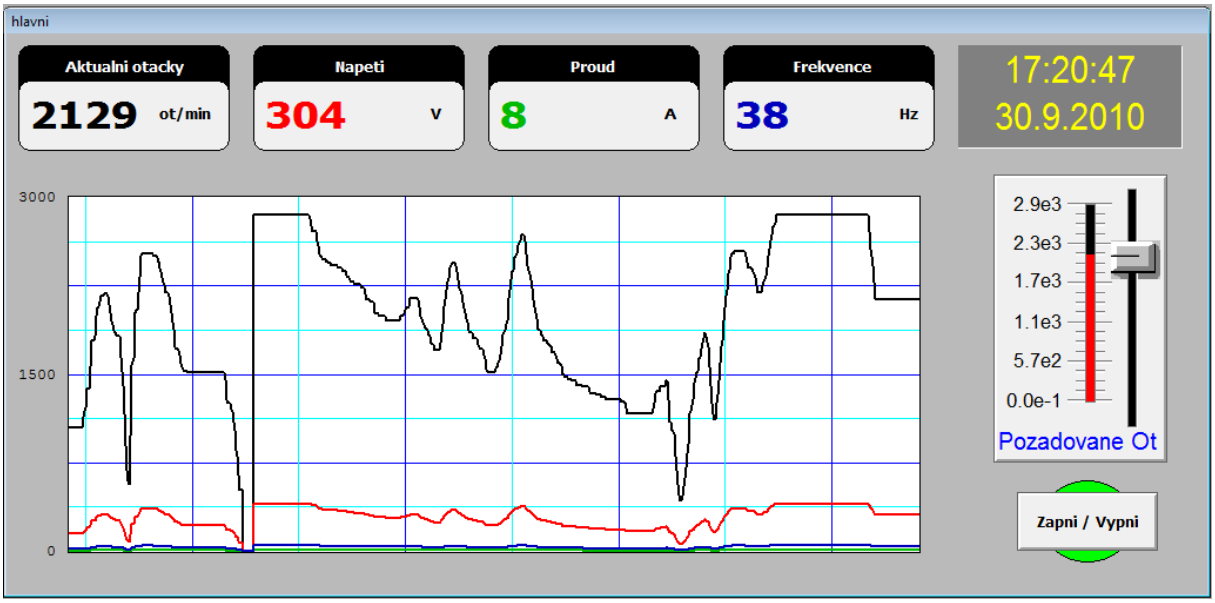
[STANÍČEK, 2009],[*Wonderware® FactorySuite™ SPCPro™*, 1999]

6 InTouch aplikace

Podle zadání jsem vytvořil nejdříve aplikaci se simulovanýma veličinami. Poté jsem sestavil už plnohodnotnou aplikaci pro testování SPC nástrojů, napojenou na reálný model.

6.1 Aplikace se simulovanými veličinami

Demonstrační aplikace je sestavena v programu InTouch. Simulace veličin je zajištěna pomocí nadefinovaných skriptů. Program tedy jednoduše simuluje chování aplikace napojené na reálný model.



Obr. 6.1 – Ukázka demonstrační aplikace

Aplikace je sestavena z jednoho okna, které obsahuje čtyři měřené veličiny. Jejich barevné označení koresponduje s barvami reálného trendu umístěného v dolní části. Další součásti okna jsou také posuvník, (nastavuje požadovanou hodnotu otáček), vypínač s barevným označením (vypíná a zapíná motor) a hodiny s aktuálním datem.

Tabulka 6.1 ukazuje seznam použitých proměnných demonstrační aplikace. K simulaci měřených hodnot je použit jednoduchý přepočet, řešený v aplikačním skriptu, který běží stále v pozadí aplikace, dokud není ukončena.

Tab. 6.1 – Seznam proměnných v demonstrační aplikaci

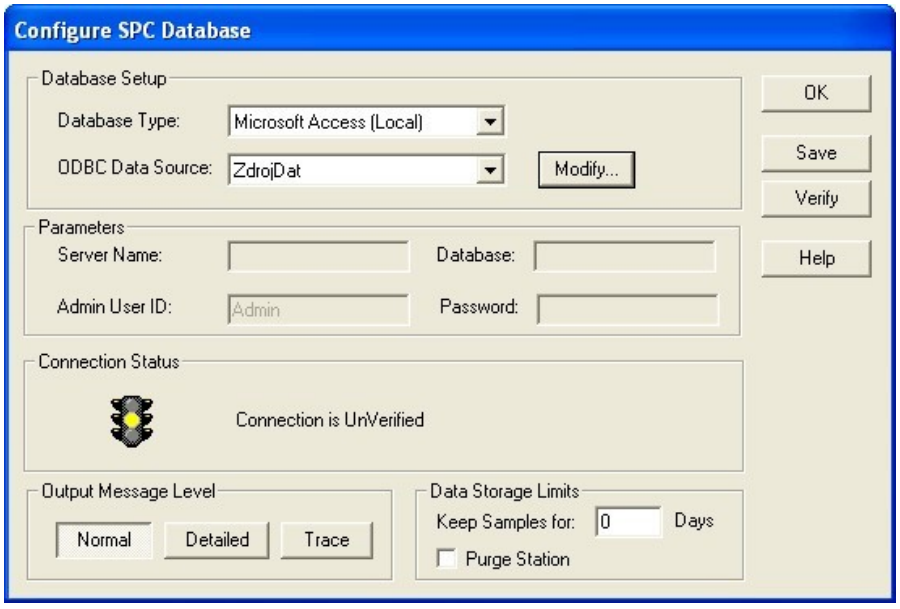
Název	Typ	Rozsah	Trend	Popis
otacky	Memory Real	0-2850	reálný	Aktuální otáčky
napeti	Memory Real	0-400	reálný	napětí
proud	Memory Real	0-10	reálný	proud
frekvence	Memory Real	0-50	reálný	frekvence
pozotacky	Memory Real	0 - 2850	-	Požadované otáčky
spust	Memory Discrete	-	-	spouštění

6.2 Aplikace s reálnými daty a SPC nástroji

Abych mohl plně otestovat funkčnost SPC nástrojů na reálném modelu, bylo nutné sestavit aplikaci, která tyto data sbírá a vizualizuje. Nejdříve je třeba SPC nástroje konfigurovat.

Konfigurace SPC nástrojů

SPC nástroje se nacházejí v aplikaci ve SPC okně. Jsou použity dva druhy a to Regulační diagram (Control chart) a Histogram. Princip těchto nástrojů je uveden v páté kapitole.

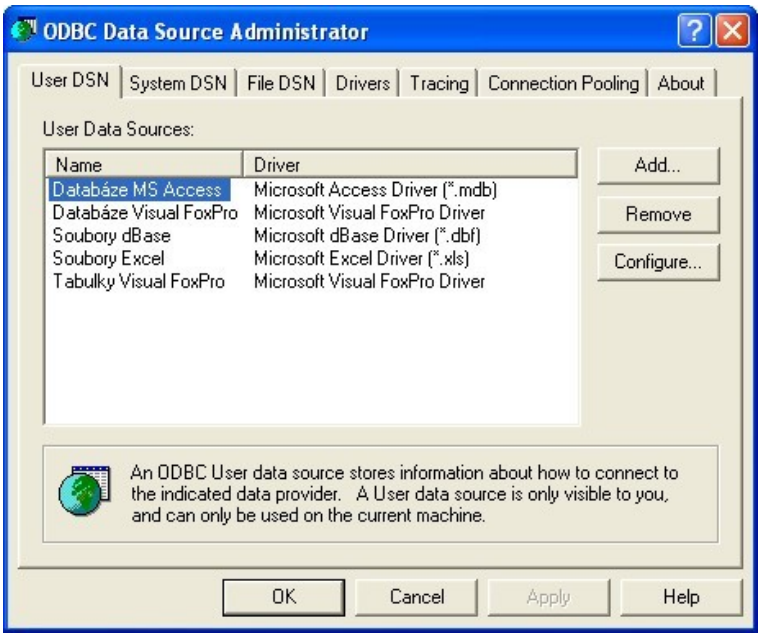


Obr. 6.2 – Okno SPC databáze

Tab. 6.2 – Seznam proměnných v aplikaci s reálnými daty

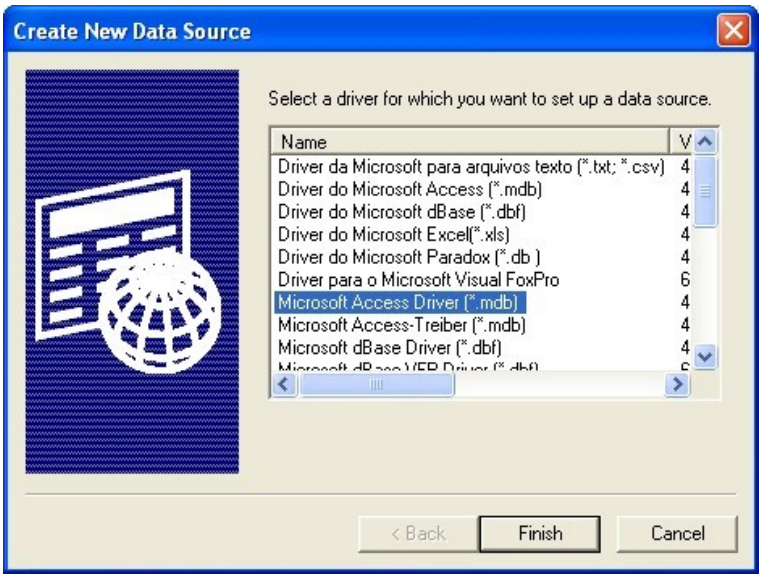
Název	Typ	Item	Rozsah	Trend	Popis
AktOtackyPREP	I/O Real	rAktualni_rychlostPREP	0 - 32767	reálný	Přepočítané aktuální otáčky
Frekvence	I/O Real	rFrekvence	0 - 32767	reálný	Aktuální frekvence
Proud	I/O Real	rProud	0 - 32767	reálný	Aktuální proud
Napeti	I/O Real	rNapeti	0 - 32767	reálný	Aktuální napětí
RefOtacky	I/O Real	rZadane_otacky	0 - 2850	-	Nastavení referenčních otáček
Start	I/O Discrete	dZapnout_motor	-	-	Spuštění motoru
Stop	I/O Discrete	dVypnout_motor	-	-	Zastavení motoru
Reset	I/O Discrete	dResetuj_chyby	-	-	Resetování chyb na měniči
Priprav	I/O Discrete	dPripravit	-	-	Připravení měniče k chodu
Sber_dat	Memory Discrete	-	-	-	Indikace sběru dat

Aby mohly SPC nástroje dobře fungovat, je třeba nejdříve vytvořit databázi. Jsou dvě možnosti, a to MS Access nebo MS SQL Server. Pokud použijeme aplikaci pro jediný uzel (můj případ), lze použít obě možnosti. Ovšem za předpokladu použití pro více uzlů, je třeba využít MS SQL Server.



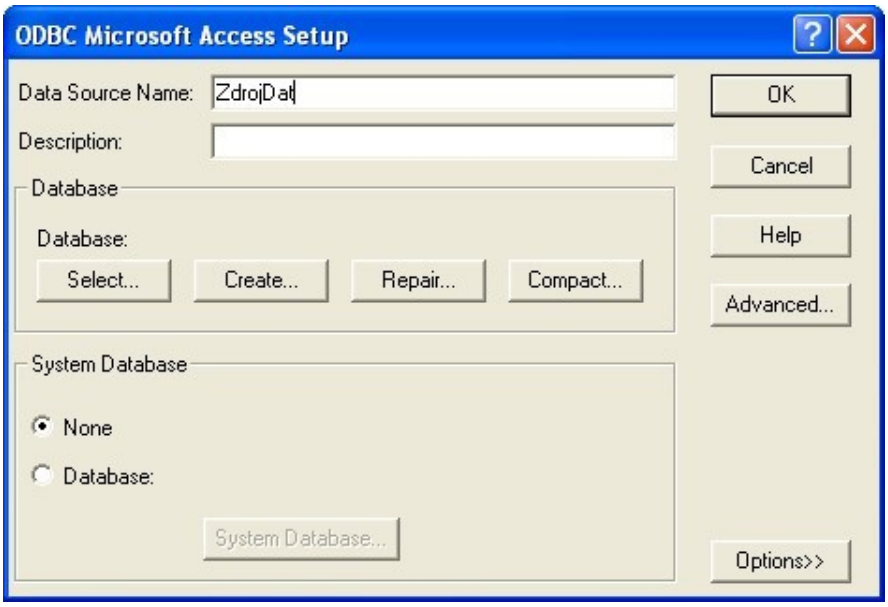
Obr. 6.3 – Okno administrace ODBC zdroje

Databáze a ODBC zdroj se vytváří a konfiguruje ve WindowsMaker v okně Configure SPC Database. V části *Database type* se zvolí *Microsoft Access (Local)* a v části *ODBC Data Source* se zvolí <*new*> aby byla založena nová databáze.



Obr. 6.4 – Okno vytvoření nového datového zdroje

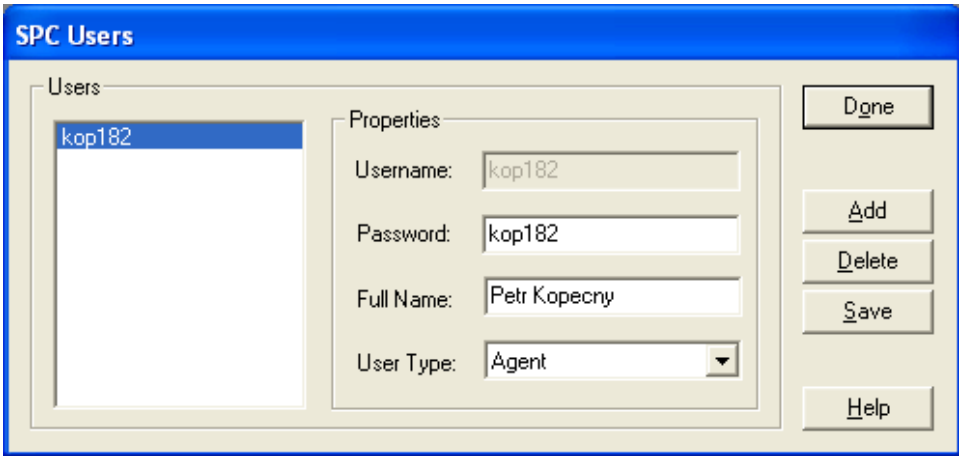
Otevře se okno ODBC Data Source Administrator kde se klikne na tlačítko *Add*. Po zobrazení okna Create New Data Source se musí označit *MS Access driver (*.mdb)* což je příslušný ovladač k databázi.



Obr. 6.5 – Okno nastavení ODBC Microsoft Access

Po stisknutí tlačítka *Finish* se dostaneme do okna OBDC Microsoft Access Setup, kde vypíšeme *Data Source Name* jméno zdroje dat. Stiskneme tlačítko *Create* a vytvoříme databázi.

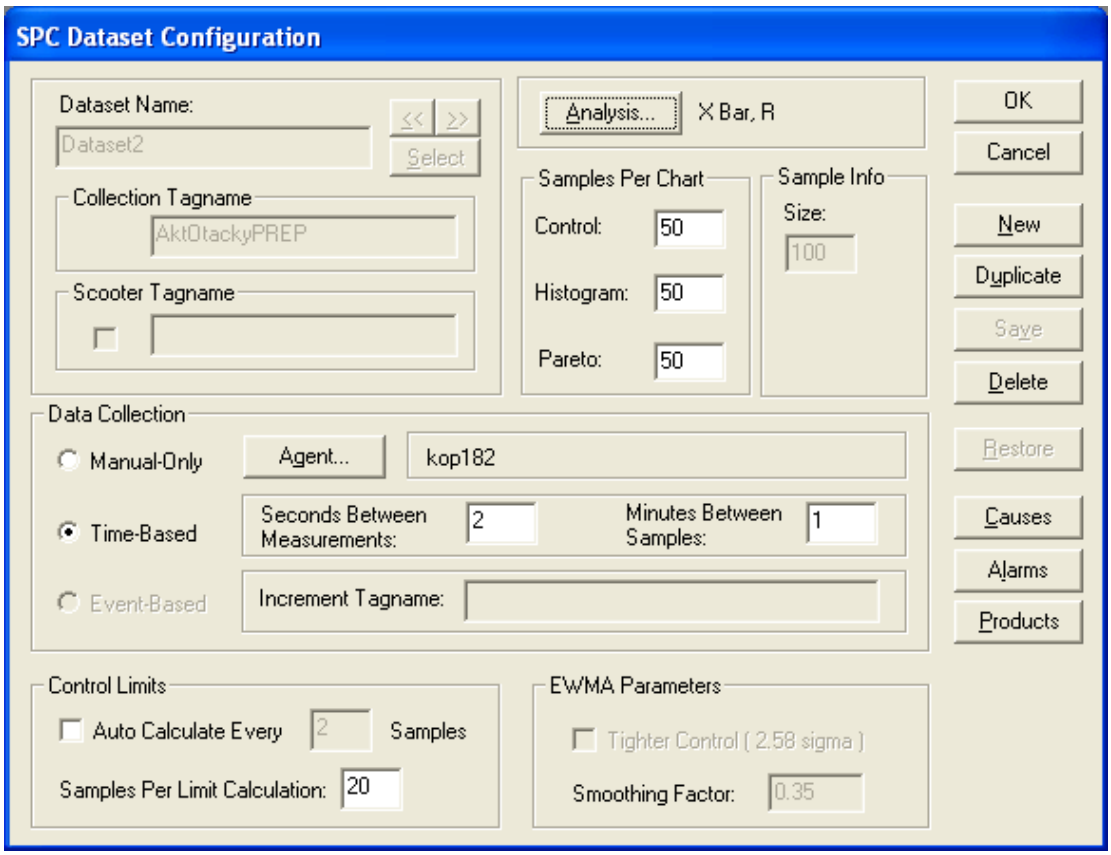
Ale jelikož databáze nebyla ještě založena je třeba vybrat umístění a jméno. Postupně odklikáme všechna dialogová okna potvrzujícím tlačítkem *OK* až k oknu Configure SPC Database.



Obr. 6.6 – Okno SPC uživatelů

Uložíme vytvořenou databázi pomocí tlačítka *Save* a po zobrazení zprávy informující o tom že naše databáze není spuštěná, pokračujeme tlačítkem *Yes* a spojení pro ODBC databázi

bude otevřeno. Potom klikneme na tlačítko *Verify*. V okně v části *Connection Status* uvidíme na semaforu zelenou, za předpokladu, že propojení bylo úspěšné.



Obr. 6.7 – Okno datasetu

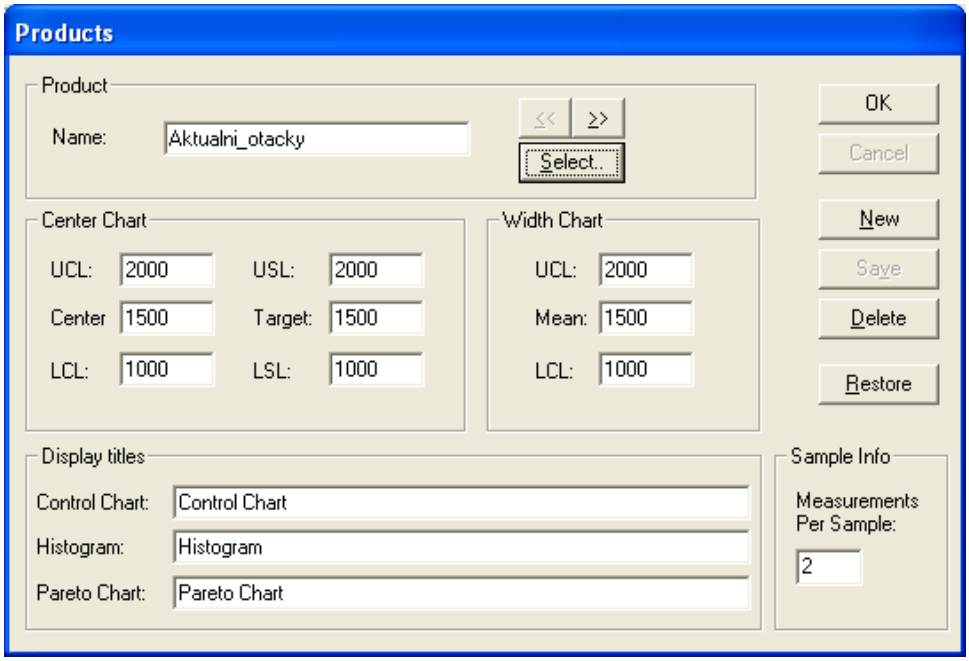
Abychom mohli použít automatické zadávání dat, musí být nakonfigurován uživatel databáze. Jinak musíme použít manuální zadávání, což je velmi nepraktické. Pro nakonfigurování uživatele je třeba otevřít dialog SPC User. Zde vypíšeme potřebné údaje, jako je jméno a heslo. Ostatní parametry jako je User Type nastavíme na Agent a ve Full Name vypíšeme plné jméno uživatele. Jméno a heslo uživatele je dále použito v tlačítku, které spouští sběr dat pro SPC a to ve formě skriptu:

```
SPCConnect("user","password");
```

Do uvozovek se dopíše jméno a heslo, v mém případě, kop182 v obou údajích. Vypínání sběru dat se pak dělá pomocí skriptu:

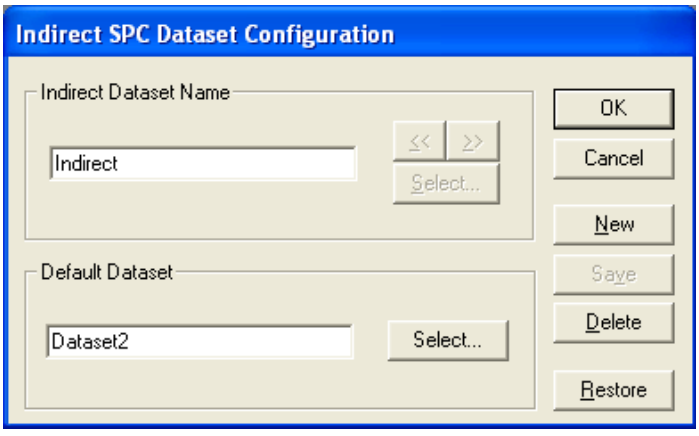
```
SPCDisconnect;
```

Aby bylo možné použít SPC nástroje, je třeba vytvořit také takzvaný Dataset nebo nepřímý dataset (Indirect dataset). Otevřeme okno SPC Dataset configuration v části SPC Dataset. Napíšeme jménu datasetu do kolonky *Dataset Name* a do kolonky *Collection Tagname* zvolíme sledovanou proměnnou (v mém případě AktOtackyPREP). Tlačítkem *Analyst* volíme typy analýzy, v části *Samples Per Chart* počet vzorků pro typ SPC nástroje.



Obr. 6.8 – Okno products

Abychom mohli nastavit limity v regulačním diagramu, musíme nastavit produkt. Do nastavovacího okna se dostaneme tlačítkem *Product*. Těchto produktů můžeme nastavit více a přepínat mezi nimi. Dále můžeme pro každý dataset nastavit alarmy a události (*Alarms*, *Causes*). [STANÍČEK, 2009]

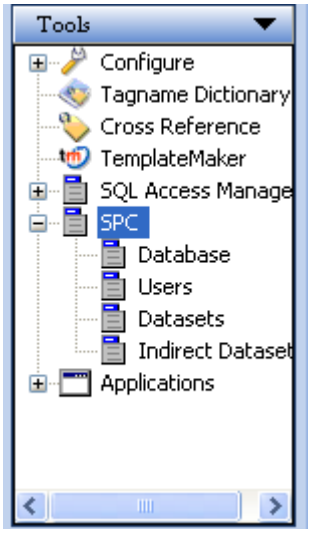


Obr. 6.9 – Okno nepřímého CPS datasetu

Další možností je nepřímý dataset (Indirect Dataset), který lze jednoduše propojit s klasickým datasetem. Stačí otevřít okno Indirect SPC Dataset Configuration pomocí SPC Indirect Dataset. Do horního pole napíšeme jméno nepřímého datasetu a ve spodním vybereme stávající dataset. Nepřímý dataset slouží k tomu, abychom mohli zobrazit ve stejném SPC grafu několik datasetů. [*InTouch HMI Supplementary Components Guide*, 2007]

SPC nástroje

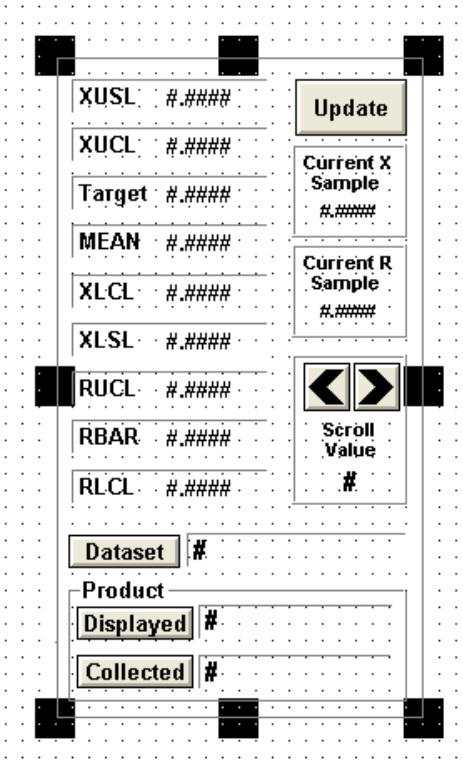
Pokud proběhlo vytvoření databáze, datasetů a produktů v pořádku, můžeme pokračovat vytvořením samotných SPC nástrojů v prostředí InTouch. Tyto nástroje najdeme v knihovně wizzard. V mém případě jsem použil, Control chart (řídící diagram), histogram a SPC limits wizzard.



Obr. 6.10 – Umístění nastavení SPC nástrojů v prostředí InTouch

Napojení Control chart a histogramu je velice jednoduché, stačí pouze vyplnit ve vlastnostech nástroje námi používaný dataset. Fungování Control chart (regulačního diagramu) je stejné, jako v kapitole 5. Jediným rozdílem je, že data se ukládají a vypočítají pomocí počítače. Histogram má navíc distribuční funkci Gaussova rozdělení a údaje o šikmosti a špičatosti.

[STANÍČEK, 2009]



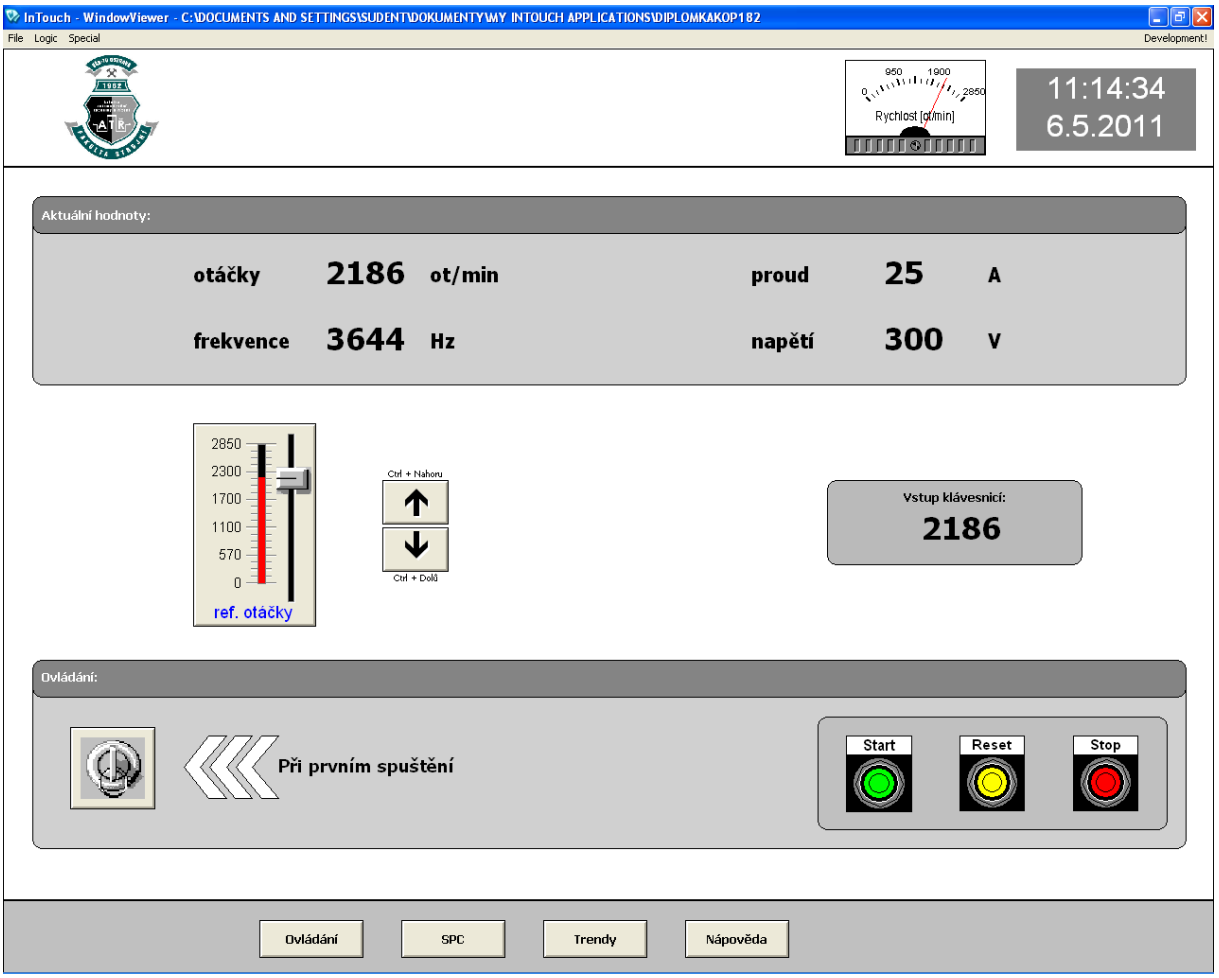
Obr. 6.11 – SPC Limits Wizzard

Horní lišta

Horní lišta funguje jako informační. Kromě ukazatele aktuální rychlosti je na ní vidět hodiny a potvrzení sběru dat SPC nástrojů, které svítí jen při aktivním sběru dat. Vlevo lze také vidět logo katedry ATŘ.

Prostřední plocha

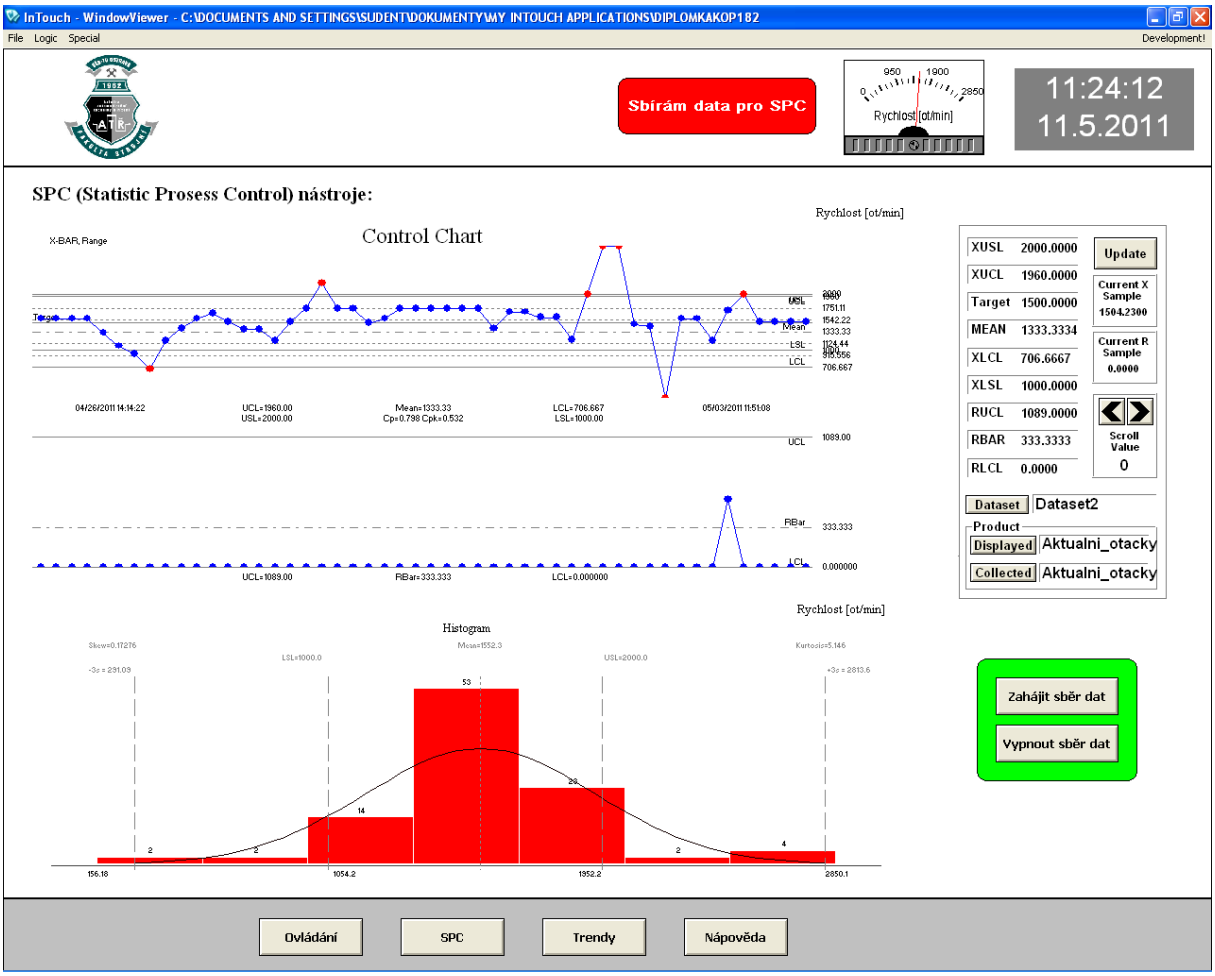
Na této ploše se přepíná několik oken určených k používání programu. Každé z oken může být zobrazeno ve stejnou chvíli pouze jediné a přepínají se pomocí tlačítek v Dolní liště. Tyto okna jsou Ovládací okno, SPC okno a Trendy.



Obr. 6.12 – Ukázka programu s reálnými daty s prostřední plochou Ovládání

1. Ovládací okno – je rozdělené do tří oblastí:
   - Aktuální hodnoty – Zde se zobrazují čtyři hodnoty čtené z PLC.
   - Ovládání otáček – Zde se dají ovládat referenční otáčky a to několika možnostmi. Klasickým posuvníkem. Vstupem klávesnice, kdy je nutné kliknout na zobrazenou hodnotu v příslušném okně a změnit ji. A tlačítky se šipkami spolu s klávesovými zkratkami.
   - Ovládání – V této oblasti jsou čtyři ovládací prvky. Prvním z nich je Klíč, kterým je třeba před spuštěním otočit a tím se vyšle signál frekvenčnímu měniči pro přípravu k funkci. Tento úkon stačí provést pouze jednou před prvním spuštěním. Tři další prvky jsou tlačítka Start, Stop a Reset. Funkce tlačítka Reset je resetování možných chyb a problémů, které mohou nastat na frekvenčním měniči. Funkce tlačítek Start a Stop je spuštění a zastavení motoru, i když jsou nastaveny referenční otáčky.

2. SPC okno – obsahuje dva SPC nástroje (Regulační diagram neboli Control chart, Histogram), tlačítko pro zapnutí nebo vypnutí sběru dat pro SPC nástroje a SPC Limits wizzard, který umožňuje nastavovat pomocí datasetů limitní hodnoty. Více o SPC níže.
3. Trendy – toto okno ukazuje čtyři průběhy čtyř sledovaných proměnných (aktuální otáčky, napětí, proud a frekvence).

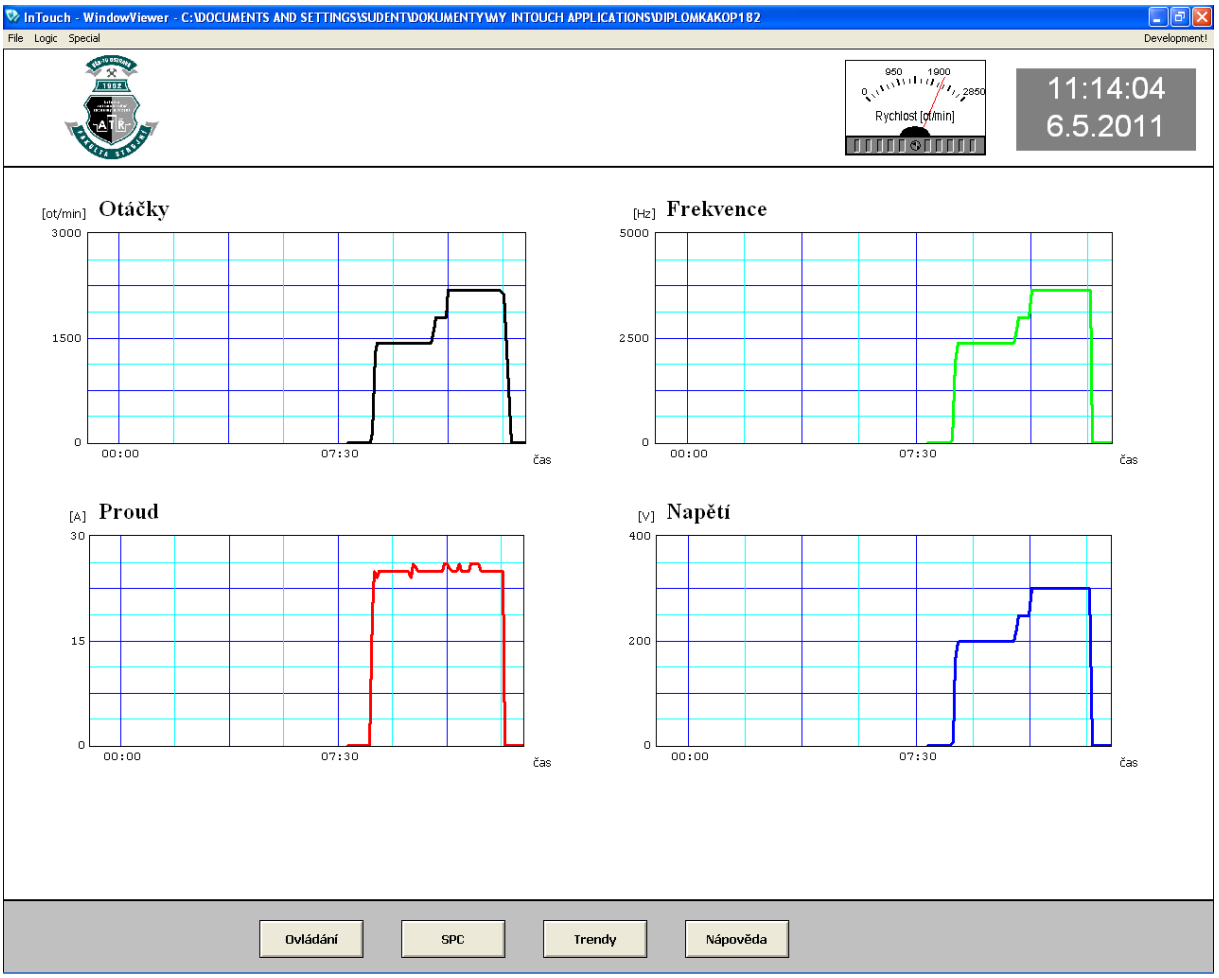


Obr. 6.13 – Ukázka programu s reálnými daty s prostřední plochou SPC

Dolní lišta

Dolní lišta tvoří funkci Menu. Je na ní umístěno několik tlačítek, které přepínají prostřední plochu na další okna. Je na ni umístěno také tlačítko pro zobrazení nápovědy s řešením jednoduchých problémů, které mohou nastat při používání programu.

Tabulka 6.2 nám ukazuje seznam proměnných. Sloupec Název ukazuje seznam proměnných ve vizualizačním prostředí InTouch. Naproti tomu, sloupec Item ukazuje propojovací slovo pro SPC Link.



Obr. 6.14 – Ukázka programu s reálnými daty s prostřední plochou Trendy

Proměnné jsou dva druhy, I/O (Input/Output, vstupně/výstupní) a Memory (Vnitřní paměťová) a dělí se v aplikaci na dva typy. Reálné (Real) a Diskrétní (Discrete). Další tři sloupce ukazují rozsah proměnných, typ připojeného trendu a krátký vysvětlující popis. Proměnná AktOtackyPREP je navíc sledována SPC nástroji.

Zhodnocení

V úvodu práce jsem rozebral koncepci vývoje aplikací v dnešní době. V podstatě existují dva druhy těchto koncepcí, a to Tag-based a Component object-based. Oba přístupy v dnešní době spolu koexistují, ovšem přístup Tag-based je už na ústupu. A to hlavně vzhledem k nižším úsporám k novějšímu Component object-based přístupu. Zvláštní výhodou naproti Tag-based přístupu je také jednoduchá a velmi efektivní úprava následných aplikací a znovupoužití předpřipravených komponent a jejich hromadná úprava.

Práce dále pokračuje popisem vybraného vizualizačního prostředí, kterým se stal InTouch společnosti Wonderware. Toto prostředí je založeno na Tag-based přístupu a zvolil jsem jej díky své jednoduchosti. Výhody poskytované Component object-based přístupem bych nevyužil a pro aplikaci této velikosti to postačuje. Ale i tak má toto prostředí velké možnosti a disponuje nástroji, jako jsou Smart symboly nebo Wizzard, které mohou tvořit velmi efektivní aplikace.

Proto, abych mohl lépe testovat SPC nástroje, byl potřeba nějaký druh reálného modelu. Tímto reálným modelem se stal motor s frekvenčním měničem. Dále propojený pomocí sběrnice Profibus k PLC zařízení na kterém byl už dříve realizovaný program. Tento program posílá a přijímá data přes Ethernet rozhraní do počítače. V počítači je navrhnuta a sestrojena aplikace v prostředí InTouch, která komunikuje pomocí OPC Serveru a OPC Linku s PLC.

Z velké škály SPC nástrojů jsem vybral Control chart (regulační diagram) a histogram. Oba tyto nástroje podporuje prostředí InTouch/SPCPro. Způsoby nasazování řídících diagramů jsem popsal na dvou typech. Regulace měřením a srovnáváním. Regulace měřením je právě ten druh, který se v SPCPro používá. Dále se rozděluje na několik typů diagramů. Tři nejdůležitější jsou Regulační diagram pro samostatné hodnoty, takzvaný X-individual, regulační diagram pro průměr a rozptyl a regulační diagram pro průměr a směrodatnou odchylku. V aplikaci napojené na reálný model jsem použil regulační diagram pro průměr a rozptyl. Tento diagram jsem také sestavil v tabulkovém procesoru MS Excel a potvrdil správnost řešení v programu Statgraphics. Regulační diagram můžeme zařadit do první skupiny metod řízení kvality, pochopení procesu. Je možné proces velmi jednoduše sledovat a případně odhalit odchylky, ale je třeba dát pozor na chyby prvního i druhého typu. Druhý SPC nástroj, histogram, jsem taktéž rozebral a připravil v MS Excel. Přidal jsem i výpočty šikmosti a křivosti, které udávají důležité informace o statistickém souboru dat.

V rámci seznamování s reálným modelem jsem také sestavil aplikaci se simulovanými veličinami. Tato aplikace nemá napojení na reálný model. Konkrétní SPC nástroje jsem poté testoval na aplikaci v prostředí InTouch přímo napojenou na reálný model, motor. Dále jsem se věnoval nastavení a vytvoření databáze, která je nutná pro SPC nástroje a jejich funkčnost.

Výsledkem této práce je již hotová vizualizační aplikace napojená na reálný model, motor. V aplikaci jsou nakonfigurovány nástroje pro monitorování a analýzu pomocí SPC nástrojů (SPCPro), které nabízí prostředí InTouch.

Protože hlavním cílem bylo vysvětlení metodiky SPC řízení spolu s ověřením složitosti výpočtů bez nasazení SPC nástrojů do praxe s použitím lépe přístupným programům, jako je tabulkový procesor MS Excel, aplikace nezohledňuje poruchy. Aplikace umožňuje jednoduchou a rychlou změnu, případně přidání sledované veličiny v její další verzi.

Použitá literatura

GARBRECHT, Steven. *The Benefits of Component Object-Based SCADA and Supervisory System Application Development*. [s.l.] : Wonderware, 2006. 16 s.

HERMANS, R. *Skewness_Statistics.svg* [online]. [s.l.] : [s.n.], 16.8.2008 [cit. 2011-05-17]. Dostupné z WWW: <http://upload.wikimedia.org/wikipedia/commons/b/b3/Skewness_Statistics.svg>.

InTouch HMI Supplementary Components Guide [online]. [s.l.] : Invensys Systems, 6.6.2007 [cit. 2011-05-17]. Dostupné z WWW: <http://www.logic-control.com/media/IT10Supplementary.pdf>.

MELOUN, M.; MILITKÝ, J. *Statistické zpracování experimentálních dat*. Praha : Plus, 1994. 839 s. ISBN 80-85297-56-6.

MILITKÝ, Jiří; KŘEMENÁKOVÁ, Dana. *Techniky řízení jakosti s aplikací v textilu* [online]. Liberec : Technická univerzita v Liberci, 22.2.2000 [cit. 2011-05-17]. Dostupné z WWW: <http://www.ft.vslib.cz/depart/ktm/files/20060106/japis.pdf>.

NIST/SEMATECH e-Handbook of Statistical Methods [online]. 6/01/2003 [cit. 2011-05-17]. Dostupné z WWW: <http://www.itl.nist.gov/div898/handbook/>.

NOSKIEVIČOVÁ, Darja. *Statistické metody v řízení jakosti*. Ostrava : VŠB - Technická univerzita Ostrava, 1996. 81 s. ISBN 80-7078-318-4.

OSADNÍK, Petr. *Modelování průmyslových objektů v prostředí průmyslového aplikačního serveru*. Ostrava, 2008. 73 s. Diplomová práce. VŠB-Technická univerzita Ostrava, Fakulta strojní, Katedra Automatizační techniky a řízení.

Pantek : výrobní inteligence v průmyslové automatizaci [online]. 2009 [cit. 2011-05-17]. Dostupné z WWW: <http://www.pantek.cz/>.

POKORNÝ, M.; KOZUB, R. *Statistické zpracování měřených dat I.*. Ostrava : VŠB - Technická univerzita Ostrava, 1998. 134 s. ISBN 80-7078-493-8.

Popis demonstračního zařízení vybaveného měničem kmitočtu ACS 800 : Základní přehled. [s.l.] : ABB, 29.2.2008. 36 s.

PŘIDAL, Petr. *Sledování a řízení kvality výrobních procesů*. Ostrava, 2007. 21 s. Semestrální práce. VŠB-Technická univerzita Ostrava, Fakulta strojní, Katedra Automatizační techniky a řízení.

ŘÍHA, Pavel. Intellution Plant Intelligence Solutions . *Automa : časopis pro automatizační techniku* [online]. 2002, 10, [cit. 2011-05-17]. Dostupný z WWW: <http://www.odbornecasopisy.cz/index.php?id_document=28588>.

STANÍČEK, Petr. *Monitorování a řízení kvality výrobních procesů*. Ostrava, 2009. 46 s. Diplomová práce. VŠB-Technická univerzita Ostrava, Fakulta strojní, Katedra Automatizační techniky a řízení.

Statgraphics Centurion : data analysis and statistical software [online].StatPoint Technologies, ©2011 [cit. 2011-05-17]. Dostupné z WWW: <http://www.statgraphics.com/index.htm>.

SWEEP, M. *Standard_symmetric_pdfs.png* [online]. [s.l.] : [s.n.], 6.12.2006 [cit. 2011-05-17]. Dostupné z WWW: <http://upload.wikimedia.org/wikipedia/commons/e/e6/Standard_symmetric_pdfs.png>.

The Michigan Chemical Process Dynamics and Controls Open Text Book [online]. 2009 [cit. 2011-05-17]. Control Chart Constants. Dostupné z WWW: <https://controls.engin.umich.edu/wiki/index.php/Control_Chart_Constants>.

TOMANEK, Jiří. *Vizualizace v průmyslové automatizaci*. Ostrava, 2006. 42 s. Bakalářská práce. VŠB-Technická univerzita Ostrava, Fakulta strojní, Katedra Automatizační techniky a řízení.

VLACH, Jaroslav. *Řízení a vizualizace technologických procesů*. 1. vydání. Praha : BEN - Technická literatura, 1999. 160 s. ISBN 80-86056-66-X.

Wonderware® FactorySuite™ SPCPro™: User's Guide [online]. Revision C. Irvine, CA : Wonderware Corporation, 1999 [cit. 2011-05-17]. Dostupné z WWW: <ftp://ftp.enm.com/Support/Wonderware/Intouch/WW_FS2K_71_INTOU%20%28D%29/InTouch/userdocs/intouchspc.pdf>.

ZAVADIL, Jaroslav. *Funkce a ovládání modelu asynchronního motoru a frekvenčního měniče.* Ústní sdělení. Ostrava, 2010-05-12.

ZAVADIL, Jaroslav. *Systém monitorování a řízení technologie*. Ostrava, 2009. 57 s. Diplomová práce. VŠB-Technická univerzita Ostrava, Fakulta strojní, Katedra Automatizační techniky a řízení.